

HACKER

Season 06 / Issue 01
τεύχος 061 - Striking Simplicity Edition

Δημιουργία static site με το Hugo [*]

Διαχείριση εικονικών τοπικών δικτύων στον κόσμο του libvirt

Αυτόματο deployment Hugo site, μέσω Git Hooks

Υπηρεσία DNS με έμφαση την ασφάλεια

Εκκίνηση SystemRescueCd από PXE server

Proxy commands για το SSH και SOCKS tunneling

[*] γιατί παρά τις διαβεβαιώσεις περί του αντιθέτου, Less is More

HACKEЯ


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

#dH061 | Περιεχόμενα

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Γιώργος Γιάννου, Γιάννης Κρομμύδας, Πέτρος Κυλαδίτης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ψηφιακό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations. Ο δικτυακός τόπος του περιοδικού είναι στο https://deltahacker.gr και το email επικοινωνίας είναι το talk2us@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο https://deltahacker.gr/order κι από την ίδια σελίδα γίνονται οι παραγγελίες.

Κάτι απλό, που έχει τον ατελείωτο

του Χρήστου Βαρελά

Φίλες και φίλοι,

- Καλός συνάδελφος και φίλος, δύο και κάτι μήνες πριν, είχε προσπαθήσει να με προειδοποιήσει.
- *Ωραίο προτζεκτάκι, ξεκίνησέ το, μου είχε πει, αλλά μην ασχοληθείς πάρα πολύ. –Και γιατί να μην ασχοληθώ πάρα πολύ; –Γιατί μετά παίζει να μη σου κάνει καρδιά να τ' αφήσεις και να προχωρήσεις σε κάτι άλλο, πιο σύνθετο ή έστω πιο ενδιαφέρον.*
- Έλεγα στο συνάδελφο για το qboxes (https://github.com/colder-is-better/qboxes), ένα script σε BASH που σκεφτόμουν να σκαρώσω ώστε να διευκολύνομαι λίγο με τη δουλειά στη δουλειά.
- Διόρθωση: για να διευκολύνομαι λίγο με τη δουλειά στη νάιν-του-φάιβ, όπως συνήθιζαν να λένε και οι εντρεπρεν(έ|ού)ρς. Ίσως ακόμη να λένε "νάιν του φάιβ" –ειδικά όσοι βρίσκονται ακρός δε ποντ– αλλά κάτι μου λέει πως οι της απ' εδώ μεριάς (του ποντ) έχουν κάπως ξενερώσει – ή έστω απλά ηρεμήσει.
- Το qboxes που λέτε, είναι ένα σκριπτ που πλέον χρησιμοποιώ συχνά-πυκνά, ώστε ξεκινώντας από το μηδέν να συγκροτώ libvirt-based Ceph clusters για installation/regression testing, αλλά και για να δοκιμάζω με ασφάλεια όποια ιδέα μου έρχεται στο μυαλό (όποτε μου έρχεται).
- Το qboxes δεν κάνει κάτι που δεν μπορούν να κάνουν εργαλεία σαν τα virsh (https://linux.die.net/man/1/virsh), virt-clone (https://linux.die.net/man/1/virt-clone), virt-customize (http://libguestfs.org/virt-customize.1.html) κ.ά., είναι όμως αρκετά **πιο απλό** στη χρήση και σκόπιμα προσαρμοσμένο στο δικό μου γουόρκ-φλόου. Ακριβέστερα: Κάνει λιγότερα σε σχέση με όσα μπορούν να κάνουν εργαλεία από τα πακέτα libvirt-client, guestfs-tools και virt-install, αλλά μαθαίνεται πιο εύκολα και είναι περισσότερο επεξηγηματικό σε περίπτωση που κάτι δεν πάει καλά.

- Σχετικά γρήγορα, το σκριπτάκι άρχισε να με βολεύει και να με διευκολύνει στην πράξη – κι όχι μόνον όσον αφορά στο Ceph (https://ceph.com). Από νωρίς μάλιστα είχα αρχίσει να το διασκεδάζω κιόλας. Πώς να το κάνουμε, άλλο είναι να συντάσσεις μια αναφορά και στην ενότητα regression testing plan να παραθέτεις εντολές με λίγο-πολύ γνωστά εργαλεία, κι άλλο να αμολάς –ξαφνικά και χωρίς καμία πρότερη προειδοποίηση– εντολές όπως `qboxes define --ram=2048 --cores=2 --volume-size=256 --network=qbxnet --guest-os=sles12sp3 --pool=ses5 osd{1..5}` – και στο καπάκι να προσθέτεις κι ένα `qboxes attach --name=data --size=2048 --pool=tank osd{1..5}`. (Εντάξει, θα μπορούσα να είχα παραθέσει λίγο απλούστερα παραδείγματα. Πότε πότε όμως θέλω κι εγώ να τραβάω την προσοχή σας. Καταλαβαίνετε.)
- Δύο μήνες πριν, λοιπόν, το qboxes ήταν κατά κύριο λόγο σχεδιασμένο για έναν χρήστη, υπό την έννοια ότι δεν με ένοιαζε και πολύ αν κάποιος άλλος θα το έβρισκε χρήσιμο ή όχι. Αυτό τουλάχιστον έλεγα στον εαυτό μου, έως τη στιγμή που ένας –ψέματα, 100% περισσότεροι– συνάδελφοι το πρόσεξαν, κι έριξαν μια ματιά. Μετά από μερικές κουβέντες που ακολούθησαν, ομολογώ πως άρχισα να βλέπω το qboxes διαφορετικά.
- Η κεντρική ιδέα είναι ότι ναι μεν αποτελεί ένα wrapper script άλλων εργαλείων που σχεδιάστηκε για να διευκολύνει τον γράφοντα, αλλά με συγκεκριμένες τροποποιήσεις και προσθήκες ίσως αποδειχθεί ενδιαφέρον σε περισσότερους από έναν χρήστες.
- Επιτρέψτε μου να παραθέσω δύο παραδείγματα, για να σας δείξω προς τα πού το πάω. Γενικά, το qboxes καταλαβαίνει μια σειρά από εντολές, κι αυτές με τη σειρά τους δέχονται άλλες υποεντολές ή/και παραμέτρους.
- Μία από τις εντολές που καταλαβαίνει το qboxes είναι η annihilate. Μπορούμε να πούμε ότι είναι αντίστοιχη της εντολής undefine, που καταλαβαίνει το virsh, αλλά κάνει πολλά περισσότερα. Αναλυτικότερα, πληκτρολογώντας `virsh undefine guest` διαγράφεται το αρχείο XML που περιγράφει την εικονική μηχανή με όνομα guest, όμως τα σχετικά storage volumes (sparse QCOW2 files) και φυσικά τα όποια snapshots παραμένουν άθικτα. Είναι ευθύνη του χρήστη να βρει που είναι αποθηκευμένα τα αντίστοιχα αρχεία και να τα διαγράψει, αν φυσικά δεν τα θέλει. Ο χρήστης μάλιστα οφείλει να είναι επιμελής όταν διαγράφει volumes, αφού αν δεν το κάνει σωστά τότε τα αντίστοιχα storage pools θα φαίνεται σαν να περιέχουν τα volumes – κι ας έχουν αυτά διαγραφεί. Τώρα, αν κάποιος ξέρει καλά το virsh, δεν θα έχει κανένα πρόβλημα να διαγράφει **ολοκληρωτικά** και **σωστά** όποια VMs θέλει. Εναλλακτικά, ο ίδιος "κάποιος" μπορεί να καταφύγει στο qboxes και να πληκτρολογήσει μία

μόνο εντολή, σαν αυτή: `qboxes annihilate guest`. Τίποτε περισσότερο. Το εργαλείο θα εξαφανίσει το XML της μηχανής, τα όποια snapshots έχει, το ίδιο και τα storage volumes της. Θα ενημερώσει, επίσης, και τα αντίστοιχα storage pools, ώστε να μην εμπεριέχουν άκυρα metadata για τα volumes. Σκέφτομαι, τώρα: η εντολή annihilate αφορά σε VMs και μόνο. Μήπως αξίζει να αλλάξω το command structure του qboxes, ώστε η annihilate να διαγράφει και storage pools, storage volumes ή ακόμη και libvirt networks;

- Δεύτερο παράδειγμα, αυτή τη φορά με την εντολή create, για τη δημιουργία είτε νέων storage pools είτε νέων libvirt networks. Σοβαρά τώρα, το βρίσκετε εσείς σωστό για μια εντολή που λέγεται create, να μην είναι ικανή για τη δημιουργία νέων εικονικών μηχανών ή νέων volumes; Μέχρι στιγμής, για τη δημιουργία ενός VM μέσω του qboxes υπάρχει η εντολή clone, η οποία προϋποθέτει την ύπαρξη κάποιου άλλου VM το οποίο δημιουργήθηκε, π.χ., με το Virtual Machine Manager, και τώρα χρησιμοποιείται ως template. Υπάρχει επίσης και η εντολή define, για τη δημιουργία "κενών" εικονικών μηχανών, δηλαδή VM με "άδεια" storage volumes και χωρίς guest OS. Υποθέστε ότι έχουμε ένα QCOW2 file με μια πλήρη εγκατάσταση ενός λειτουργικού. Δεν θα ήταν βολικό αν, με χρήση της create, χτίζαμε ένα πλήρες VM γύρω από το συγκεκριμένο QCOW2;
- Θα μπορούσα να συνεχίσω με παραδείγματα υποψηφίων features για το qboxes, όμως δεν χρειάζεται. Το θέμα εδώ είναι ότι ένα σχετικά απλό script, που θα έπρεπε να έχω έτοιμο και feature-complete σε μία –το πολύ δύο– εβδομάδες, φαίνεται τελικά ότι έχει τον ατελείωτο.
- Χρειάζεται ψυχραιμία και σύνεση, φίλες και φίλοι. Κάποια στιγμή ο γράφων πρέπει να φανεί γενναίος και να βάλει μια τελεία – ή έστω μια έντονη άνω τελεία. Εξάλλου υπάρχουν κι άλλα πρότζεκτ στον ορίζοντα, πολύ πιο ενδιαφέροντα. Μερικά από αυτά δεν θα είναι ριζικά διαφορετικά από το qboxes, θα έχουν όμως άλλη αξία για τον γράφοντα.
- Πώς να το κάνουμε, άλλο είναι να γράφεις σε BASH, κι άλλο σε Python 3. Επίσης: Καλό το Command Line Interface (CLI), καλύτερο όμως ένα CLI μαζί με ένα μοντέρνο web dashboard παραδίπλα.
- Ομολογουμένως, δεν είναι και πολλές οι προκλήσεις που προδιαγράφονται για το κοντινό μέλλον. Είναι όμως εξαιρετικά ενδιαφέρουσες, βεβαίως και διασκεδαστικές.



CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web



ονέιλ μέιλ

Αγαπητοί μου,

Αποκριές που είναι, σας έχω ένα μικρό προβληματάκι να το προσπαθήσουμε λίγο, έτσι για προθέρμανση.

Here it goes. Το αντιγράφω όπως το βρήκα γραμμένο.

"Σ' ένα πλοιάριο έχουν φορτωθεί 26 πρόβατα και 10 κατσίκες. Τα ζώα, μαζί με τον βοσκό που τα συνοδεύει, περιμένουν για να τα περάσουν απέναντι. Κατά τα άλλα, το πλοιάριο είναι άδειο."

"Σε λίγο ανεβαίνει και ο καπετάνιος, τσεκάρει τα ζώα και τον βοσκό, και βάζει μπρος την μηχανή. Πόσων χρονών είναι ο καπετάνιος;"

Αυτό.

Όχι, δεν σας κάνω πλάκα. Αυτό: Πόσων χρονών είναι ο καπετάνιος;

Εδώ σε θέλω κάβουρα.

Ξυπνάω το πρωί, που λέτε, και λέω ας χαζέψω λίγο τις ειδήσεις προτού ξεκινήσω να σας γράφω.

Ιράν, Novartis, Μακεδονία, Elon Musk, Κιμ Γιονγκ Ουν. Αποκριές στον Τύρναβο. Καρναβάλι στο Ρίο. Χιονοπτώσεις στη Μόσχα. Παγωνιά στη Σεούλ.

Τα γνωστά.

Έψαχνα θέματα. Με είχε ειδοποιήσει από προ ημερών ο Κύριος Εκδότης ότι επίκειται τεύχος, και είχα κατεβάσει τη γραφομηχανή από τη ναφθαλίνη.

Θα μου πείτε, στη ναφθαλίνη την βάζεις εσύ την γραφομηχανή; Εμ, ναι, ναφθαλίνη. Τι να κάνω.

Δεν θέλω να την αφήσω να σκονίζεται και η μοναδική πλαστική σακούλα που μας έχει μείνει στο σπίτι μυρίζει ναφθαλίνη. Τη χρησιμοποιούσε η μητέρα της Εξαδέλφης για τα μωρουδιακά του ανιψιού. Για κάποιο λόγο είχε αποφασίσει να τα κρατήσει -- don't ask. Την είχε φουλάρει στη ναφθαλίνη, για να μην τα φάει ο σκώρος. Και ναι μεν δεν πλησίασε σκώρος, αλλά το καημένο το πλαστικό έχει ποτίσει.

Τέλος πάντων. Διεθνής επικαιρότητα.

Στην Κίνα, λέει, έχει ξεκινήσει έντονη συζήτηση για το επίπεδο δυσκολίας των θεμάτων στις σχολικές εξετάσεις. Αφορμή ήταν το κατά τα φαινόμενα άλυτο πρόβλημα που δόθηκε σε μαθητές της πέμπτης τάξης του Δημοτικού, κάπου στην επαρχία Σουνκίνγκ.

Κάτι με ένα καΐκι, λέει, και κάτι ζωντανά που έπρεπε να περαστούν απέναντι.

Ναι, σωστά καταλάβετε, αναφέρομαι στο πρόβλημα που σας παρέθεσα παραπάνω.

Δέκα κατσίκια, εικοσιέξι πρόβατα, κι ένας βοσκός. Πόσων χρονών είναι ο καπετάνιος;

What?

Τι θα πει, πόσων χρονών είναι ο καπετάνιος; Πλάκα με κάνετε (sic); Και πού θέλετε να ξέρω πόσων χρονών είναι; Τι να κάνω; Να προσθέσω τα αρνιά και τα κατσίκια; Σοβαρολογούμε;

Σύμφωνα, λοιπόν, με το BBC (όπου βρήκα τη συγκεκριμένη είδηση), η τοπική επιθεώρηση δημοτικής εκπαίδευσης εξηγεί πως ερωτήσεις σαν αυτή έχουν ως στόχο να οξύνουν το κριτικό πνεύμα των εξεταζομένων και να δοκιμάσουν τη δυνατότητά τους να σκέφτονται δημιουργικά. Το πρόβλημα μοιάζει σε μας ακατανόητο, αλλά ο μέσος μαθητής Πέμπτης δημοτικού στην επαρχία Σουνκίνγκ έχει στη διάθεσή του όλες τις απαραίτητες πληροφορίες για να το απαντήσει.

Τώρα να σας πω τον πόνο μου, δεν ξέρω πόσο οξύ ή δημιουργικό είναι το πνεύμα των μαθητών δημοτικού στην επαρχία Σουνκίνγκ, αλλά ευγνωμονώ τη μοίρα που δεν είμαι συμμαθητής τους. Όχι τίποτα άλλο, θα ήμουν ακόμα στην Πέμπτη Δημοτικού πασχίζοντας κι εγώ να περάσω απέναντι σαν τα αρνιά και τα κατσίκια.

Βέβαια, είμαι ελαφρώς άδικος με τον εαυτό μου.

Γιατί αν ήμουν πράγματι μαθητής δημοτικού στην επαρχία Σουνκίνγκ, δεν θα δυσκολευόμουν να βρω πως το συνολικό βάρος των 36 ζώων (26 πρόβατα και 10 κατσίκες) είναι γύρω στους εφτάμιση τόνους. Θα μπορούσα να το υπολογίσω χρησιμοποιώντας τις γνώσεις μου από το βιβλίο της ζωολογίας Πέμπτης Δημοτικού (Κίνας).

Ούτε θα χρειαζόταν να έχω αποστηθίσει το βιβλίο της αγωγής του πολίτη για να γνωρίζω πως για να οδηγήσεις πλοιάριο που φέρει φορτίο χρειάζεσαι δίπλωμα. Λίγο να είχα διαβάσει θα το ήξερα πως, για να πάρεις δίπλωμα καπετάνιου για πλοιάριο με φορτίο, πρέπει να είσαι τουλάχιστον 23.

Το πόσο διαβασμένος είμαι θα φαινόταν ίσως από το αν μπορούσα να θυμηθώ πως για πλοία με φορτίο άνω των πέντε τόνων θα πρέπει να έχεις το δίπλωμά σου ήδη εδώ και πέντε χρόνια.

Πού μας οδηγούν όλα αυτά;

Μα πού αλλού, παρά στη λύση του προβλήματός μας -- λύση ηλίου φαεινότερη στους οξύνοες και δημιουργικούς μαθητές από την επαρχία Σουνκίνγκ. Ο καπετάνιος ενός πλοιαρίου με 36 ζωντανά δεν μπορεί να είναι κάτω από 28.

Έτσι που λέτε με την Κίνα. Αλλά το θέμα μου σήμερα δεν είναι η Κίνα.

Η συγκεκριμένη είδηση μου φάνηκε ενδιαφέρουσα για έναν επιπλέον λόγο.

Ας ξαναδούμε την ερώτηση. Σε μας που δεν είμαστε από την Σουνκίνγκ, η εκφώνηση μοιάζει ακατανόητη. Τι σχέση μπορεί να έχουν τα κατσίκια και τα πρόβατα με την ηλικία του καπετάνιου; Καμία. Δεν υπάρχει τρόπος να απαντήσουμε στο πρόβλημα όσο και να προσπαθήσουμε. Και αυτό γιατί απαιτείται άμεση, αυθόρμητη πρόσβαση σε μια σειρά από τοπικά διοικητικά δεδομένα, π.χ., πότε παίρνει κανείς δίπλωμα πλοιαρίου, ποιες είναι οι διαβαθμίσεις επιτρεπτού φορτίου ανά τύπο διπλώματος κ.λπ. Χρειάζομαι επίσης και μια σειρά από άλλες ψιλο-στάνταρ πληροφορίες, όπως είναι το βάρος του μέσου αρνιού ή της μέσης κατσίκας.

Με άλλα λόγια, η συγκεκριμένη ερώτηση δεν είναι "ουδέτερη". Βασίζεται σε πληροφορίες που δεν περιλαμβάνονται στην εκφώνηση, αλλά που αναμένεται να γνωρίζεις. Σε μας το πρόβλημα φαίνεται άλυτο, αλλά στους μαθητές της Σουνκίνγκ όχι αναγκαστικά.

Πιο ουδέτερη θα ήταν μια ερώτηση σαν την παρακάτω:

"Πόσο κάνει 7 + 5;"



Πράγματι. Αν ξέρουμε αριθμητική μπορούμε να την απαντήσουμε όπου και να είμαστε, στην επαρχία Σουνκίνγκ, στην Κουάλα Λουμπούρ ή στα Ψαχνά Ευβοίας.

Βέβαια, ακόμα και εδώ προϋποθέτουμε πως αυτός που θα την διαβάσει θα πρέπει να μπορεί να την καταλάβει στα Ελληνικά. Οπότε ας την διεθνοποιήσουμε λίγο περισσότερο.

"7 + 5 = ?"

ΟΚ. Ετσι είναι καλύτερα. Ας την ονομάσουμε (q).

Μιλάμε λοιπόν για την ερώτηση (q).

Multiple Choice: Ποια από τις παρακάτω τέσσερις απαντήσεις είναι η σωστή;

(α) 12

(β) 13

(γ) 14

(δ) C

Ελα να βλέπω τα κομπιουτεράκια! Τι σας βγάζει;

Δώδεκα; Είστε σίγουροι; Δεν έβγαλε 13;

Όχι 13; Μήπως έβγαλε 14;

Μήπως έβγαλε C και δεν το προσέξατε;

Όχι, αγαπητοί μου, δεν προσπαθώ να σας μπερδέψω. Υποθέτω πως βλέπετε πού το πάω.

Το πάω στο ότι ακόμα και η (q), αυτή η απλούστατη και όσο πιο ουδέτερη γίνεται ερώτηση, έχει απάντηση που εξαρτάται από κάτι που προϋποτίθεται χωρίς να διευκρινίζεται. Εξαρτάται από τη βάση του αριθμητικού συστήματος στο οποίο έχει διατυπωθεί. Δεκαδικό; Δεκαεξαδικό; Οκταδικό; Εννιαδικό;

Βέβαια για να ακριβολογήσουμε, δεν είναι η ίδια η απάντηση που εξαρτάται από το σύστημα, αλλά το πώς θα τη γράψουμε. Η απάντηση δεν **είναι** 12. **Γράφεται** 12, στο δεκαδικό σύστημα.

Στο δεκαεξαδικό γράφεται C.

Υποερώτησις: Πώς γράφεται στο οκταδικό; Στο εννιαδικό;

Μάλιστα αξίζει να σημειωθεί πως σε κάποια άλλα συστήματα, π.χ., στο δυαδικό ή στο πενταδικό, η (q) δεν είναι καν έγκυρη, καθώς στα συστήματα εκείνα τα ψηφία "7" και "5" είναι κενά νοήματος.

Συνοψίζω.

Είδαμε πως το προβληματάκι με τα 36 ζωντανά και τον Κινέζο καπετάνιο δεν είναι και τόσο ουδέτερο, καθώς δεν μπορεί να απαντηθεί αν δεν μετέχεις έστω και λίγο της κινεζικής παιδείας (της επαρχίας Σουνκίνγκ).

Είδαμε πως ούτε η ερώτηση (q) είναι ουδέτερη, καθώς και αυτή δεν μπορεί να απαντηθεί κατά μονοσήμαντο τρόπο αν δεν θεωρήσουμε δεδομένο το ότι είναι διατυπωμένη στο δεκαδικό σύστημα -- το οποίο συνηθίζεται μεν στα μέρη μας αλλά δεν συνηθιζόταν, ξέρω 'γώ, στην αρχαία Βαβυλωνία.

Το ηθικό δίδαγμα σήμερα, Κυριακή μέρα, είναι πως στην πραγματικότητα είναι δύσκολο, αν όχι αδύνατο, να βρούμε κάποια ερώτηση που να είναι πραγματικά "ουδέτερη".

Κι αυτό δεν περιορίζεται στις ερωτήσεις και στα προβλήματα των εξετάσεων δημοτικού, στην Κίνα ή αλλού.

Ας γενικεύσω -- λίγο απότομα, το παραδέχομαι, αλλά ο χώρος είναι περιορισμένος:

Σνέιλ μέιλ

Κάθε ερώτηση, κάθε πρόβλημα, κάθε απάντηση, κάθε πρόταση, κάθε απόφανση, κάθε διατύπωση, κάθε φράση, κάθε κατάφαση, κάθε άρνηση --εν ολίγοις όλα, μα όλα τα προϊόντα της ανθρώπινης γλώσσας, όλα μα όλα τα **γλωσσικά ενεργήματα**-- δύνανται να φέρουν νόημα μόνο κατά το μέτρο που τα αναγνωρίζουμε ως κομμάτια ενός σχετικώς συνεκτικού συνολικού συμβολικού υπόβαθρου.

Τίποτα δεν είναι "ουδέτερο". Κάθε τι εντάσσεται σε ένα εννοιολογικό πλαίσιο από όπου και αντλεί την εγκυρότητά του.

Μιλώντας εδώ για εννοιολογικό πλαίσιο ή συμβολικό υπόβαθρο εννοώ, στην πραγματικότητα, τον κόσμο που μοιραζόμαστε όλοι εμείς που αλληλεπιδρούμε με την βοήθεια των συγκεκριμένων γλωσσικών ενεργημάτων.

Αν είμαστε μαθητές της επαρχίας Σουνκίνγκ, μοιραζόμαστε έναν κόσμο στο συμβολικό υπόβαθρο του οποίου υπάρχουν **ως γνωστά** το μέσο βάρος ενός προβάτου ή μιας κατσίκας ή το ηλικιακό όριο για να πάρεις δίπλωμα καΐκιού.

Αν είμαστε ενήμεροι για τα τεκταινόμενα στην γαλανή μας πατρίδα (TM), θα ξέρουμε πως η λέξη Novartis σηματοδοτεί κάτι παραπάνω από την φερώνυμη φαρμακευτική εταιρία.

Και πάει λέγοντας.

Αγαπητοί μου,

Θα θυμάστε, ίσως, αυτά που σας έγραφα στην τελευταία μου επιστολή.

Κατέληγα σε μία πρόταση που ομολογουμένως θα μπορούσε να χαρακτηριστεί ελαφρώς εκκεντρική.

"Η ανθρώπινη γλώσσα", έγραφα "*δεν* έχει ως βασική της λειτουργία την επικοινωνία. (...) Στην ανθρώπινη γλώσσα δεν υπάρχει ούτε μήνυμα, ούτε κώδικας, ούτε πληροφορία, τουλάχιστον όχι με τον τρόπο που ορίζονται αυτές οι έννοιες στη θεωρία της επικοινωνίας και αξιοποιούνται στη μελέτη της γλώσσας των μελισσών ή του DNA."

Θέλω να ελπίζω πως τα πράγματα αρχίζουν πλέον να ξεκαθαρίζουν λίγο.

Αν όχι, επιτρέψτε μου να το αναδιατυπώσω:

Η βασική λειτουργία της ανθρώπινης γλώσσας δεν είναι η επικοινωνία.

Χρησιμοποιούμε τη γλώσσα: συζητάμε, επιχειρηματολογούμε, γράφουμε επιστολές, εξομολογούμαστε, παραμιλάμε, κουτσομπολεύουμε, καβγαδίζουμε και λοιπά και λοιπά και λοιπά.

Ό,τι από όλα αυτά και να κάνουμε, αυτό που πρωτίστως κάνουμε είναι να επιβεβαιώνουμε ο ένας τον άλλο, ξανά και ξανά, πως μοιραζόμαστε έναν κόσμο -- τον συγκεκριμένο τούτο κόσμο στον οποίον αυτό που κάνουμε μπορεί να εκληφθεί ως φορέας κάποιου νοήματος. Το ποιο ακριβώς είναι αυτό το νόημα δεν έχει σημασία. Αυτό που έχει σημασία είναι πως όλοι εμείς συμφωνούμε στο ότι τα γλωσσικά ενεργήματα φέρουν νόημα.

Εξ ου και αυτό που λέω, πως η πρωταρχική λειτουργία της ανθρώπινης γλώσσας δεν είναι η επικοινωνία. Η πρωταρχική λειτουργία της είναι η δημιουργία ενός κόσμου. Κατά το μέτρο που αναγνωρίζουμε πως τα γλωσσικά μας ενεργήματα μπορεί να φέρουν νόημα, επιβεβαιώνουμε και καθησυχάζουμε (εαυτούς και αλλήλους) πως αυτό τον κόσμο τον μοιραζόμαστε.

Αν θέλετε να πάρετε μιαν ιδέα του σε τι πραγματικά αντιστοιχεί αυτό που γράφω, στην πιο θεμελιώδη, στην πιο βασική του μορφή, δεν έχετε παρά να παρακολουθήσετε μια μητέρα να γλυκοψιθυρίζει μικρά ανόητα **τίποτα** στο νεογέννητο μωρό της, και αυτό να της αποκρίνεται σαγηνευμένο.

Από εδώ αρχίζουν όλα.

Κι εδώ ξαναγυρίζουν όταν όλα τα άλλα καταρρέουν.

Ας κάνω μια παύση. Συνεχίζουμε την άλλη φορά.

Σας ασπάζομαι,

Θείος Ακάκιος

Skill: Intermediate
Tags: Hugo, Markdown, static site

Δημιουργία static site με το Hugo

Σε ένα blog ή στο site ενός πρότζεκτ, αυτό που κυρίως ενδιαφέρει και στο οποίο δίνεται τεράστια έμφαση είναι το περιεχόμενο. Στην πλειονότητα των περιπτώσεων είναι στατικό, αφού δεν υπάρχει βάση δεδομένων για την άντληση στοιχείων και τη δυναμική διαμόρφωσή του μέσω κάποιου web application.

Για τη δημιουργία δικτυακών τόπων με **στατικό περιεχόμενο** επιστρατεύονται οι σύγχρονοι **static site generators**, όπως το Jekyll (https://jekyllrb.com), το Pelican (https://blog.getpelican.com) και το Hugo (http://gohugo.io), με το οποίο κι ασχολούμαστε στο παρόν άρθρο. Να πούμε από τώρα ότι ο εν λόγω generator είναι γραμμένος στη γλώσσα Go (https://golang.org), ξεχωρίζει για το ταχύτατο rendering των σελίδων, ενώ διατίθεται και για όλα τα γνωστά λειτουργικά συστήματα. Γενικά, πάντως, αναλόγως του site που σκοπεύετε ν' αναπτύξετε είναι πιθανό να μη χρειαζόσαστε την πολυπλοκότητα ενός **Content Management System**, όπως, π.χ., είναι το WordPress. Προτείνουμε τότε να εξετάσετε σοβαρά την περίπτωση του static site. Ένα έξτρα όφελος που θα έχετε είναι και η αυξημένη ασφάλεια, αφού το **attack surface** ενός τέτοιου site είναι πάντοτε πολύ μικρότερο σε σχέση με εκείνο κάποιου site που βασίζεται σε CMS.

Εστιάζοντας στο Hugo, αρχικά το εγκαθιστούμε και το ρυθμίζουμε. Από εκεί και μετά, ένα οποιοδήποτε άρθρο για το site μας το γράφουμε στη γλώσσα υποσημείωσης Markdown (https://deltahacker.gr/?p=10306). Με τη βοήθεια του μικρού web server που διαθέτει το Hugo, βλέπουμε τοπικά (και μάλιστα ζωντανά) πώς ακριβώς θα φαίνεται το άρθρο στους επισκέπτες του δικτυακού μας τόπου. Αφού ικανοποιηθούμε με το αποτέλεσμα της δουλειάς μας, απλά ανεβάζουμε το άρθρο στον web server που ήδη έχουμε. Όλα αυτά τα παρουσιάζουμε στη συνέχεια, αναλυτικά και βήμα προς βήμα.

Λήψη κι εγκατάσταση

Το Hugo είναι δυνατόν να το μεταγλωττίσουμε από το source code, αρκεί ο compiler της Go να είναι διαθέσιμος για το λειτουργικό σύστημα της προτίμησής μας. Από την άλλη, pre-compiled binaries του Hugo διατίθενται για Linux, Mac OS X, FreeBSD και Windows. Καταλαβαίνετε λοιπόν ότι είναι **μάλλον απίθανο** να μπλέξετε με μεταγλωττίσεις – εμείς ούτε καν το σκεφτήκαμε. Την πλέον πρόσφατη έκδοση του Hugo, μεταγλωττισμένη για την αγαπημένη μας πλατφόρμα, την κατεβάζουμε από τη σελίδα με τα Hugo releases (https://github.com/gohugoio/hugo/releases). Το binary που παίρνουμε είναι αυτόνομο και δεν χρειάζεται να βρίσκεται εντός συγκε-

κριμένου καταλόγου. Αν πάντως έχουμε τα απαραίτητα δικαιώματα, τότε για λόγους ευκολίας προτείνεται να το μεταφέρουμε σε έναν από τους καταλόγους του PATH. Εναλλακτικά, αντί για το χειροκίνητο downloading μπορούμε να εγκαταστήσουμε το Hugo με τη βοήθεια ενός συστήματος διαχείρισης πακέτων/εφαρμογών. Αναλυτικότερα:

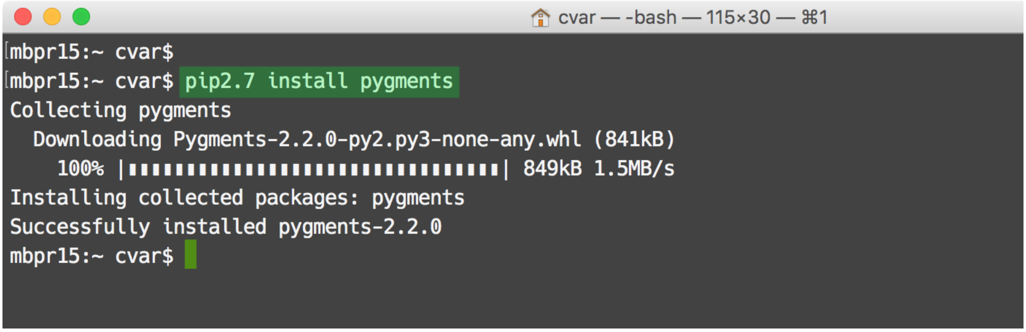
- Στο **Mac OS X** έχουμε την επιλογή του Homebrew (https://brew.sh) και η εγκατάσταση επιτυγχάνεται από ένα τερματικό, πληκτρολογώντας brew install hugo.
- Για τα **Windows** υπάρχει ο package manager ονόματι Chocolatey (https://chocolatey.org) και το Hugo το εγκαθιστούμε γράφοντας choco install hugo -confirm.
- Στο δε **Linux** το Hugo ίσως περιλαμβάνεται στα αποθετήρια της διανομής μας. Αν όμως υποστηρίζονται snaps (https://docs.snapcraft.io/core/install) τότε προτείνεται ν' ακολουθήσουμε τη συγκεκριμένη οδό (snap install hugo), ώστε να έχουμε την πλέον πρόσφατη έκδοση του Hugo.

Μετά την εγκατάσταση, από ένα τερματικό ελέγχουμε την έκδοση που διαθέτουμε πληκτρολογώντας hugo version.



Η έκδοση 0.31.1 του Hugo ήταν η πλέον πρόσφατη, τη στιγμή που γραφόταν το παρόν.

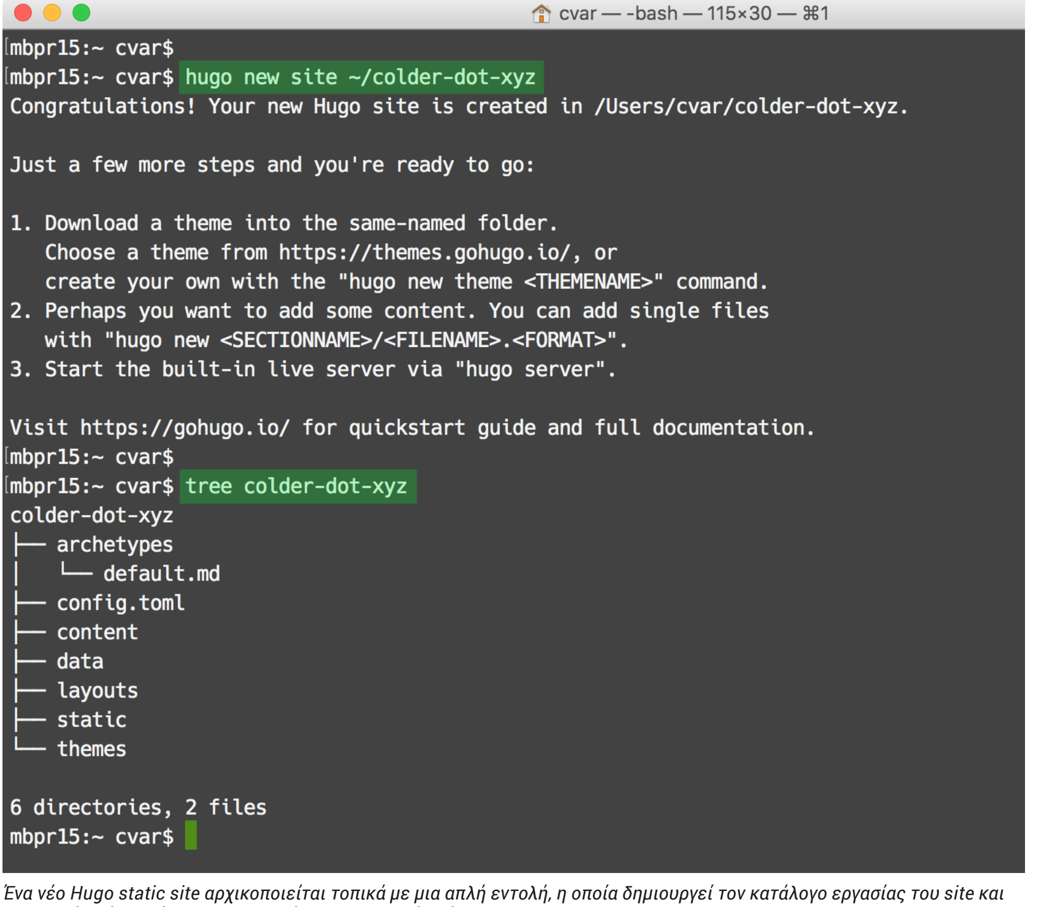
Σημείωση. Αν μας ενδιαφέρει το syntax highlighting τμημάτων κώδικα στις δημοσιεύσεις μας, τότε χρειαζόμαστε και τη μηχανή Pygments (http://pygments.org). Η εγκατάστασή της γίνεται με τη βοήθεια του εργαλείου pip της Python (π.χ., pip install pygments).



Εγκατάσταση του Pygments σε ένα σύστημα με Mac OS X, όπου η χρησιμοποιούμενη έκδοση της Python είναι η 2.7.10.

Το πρώτο μας static site με το Hugo

Ανοίγουμε ένα τερματικό και πληκτρολογούμε κάτι σαν hugo new site ~/colder-dot-xyz. Ακαριαία δημιουργείται ένας νέος κατάλογος, εντός του home directory του χρήστη μας. Το όνομά του είναι colder-dot-xyz και περιττό να πούμε ότι εσείς μπορείτε να ονομάσετε τον δικό σας όπως θέλετε. Ανεξάρτητα απ' αυτό, ρίξτε τώρα μια ματιά στα περιεχόμενά του. Θα δείτε ότι περιλαμβάνει μια δομή αρχείων και καταλόγων, η οποία χρησιμεύει για το κτίσιμο του site.



```
cvar — -bash — 115×30 — ⌘1
mbpr15:~ cvar$
mbpr15:~ cvar$ hugo new site ~/colder-dot-xyz
Congratulations! Your new Hugo site is created in /Users/cvar/colder-dot-xyz.

Just a few more steps and you're ready to go:

1. Download a theme into the same-named folder.
   Choose a theme from https://themes.gohugo.io/, or
   create your own with the "hugo new theme <THEMENAME>" command.
2. Perhaps you want to add some content. You can add single files
   with "hugo new <SECTIONNAME>/<FILENAME>.<FORMAT>".
3. Start the built-in live server via "hugo server".

Visit https://gohugo.io/ for quickstart guide and full documentation.
mbpr15:~ cvar$
mbpr15:~ cvar$ tree colder-dot-xyz
colder-dot-xyz
├── archetypes
│   └── default.md
├── config.toml
├── content
├── data
├── layouts
├── static
└── themes

6 directories, 2 files
mbpr15:~ cvar$
```

Ένα νέο Hugo static site αρχικοποιείται τοπικά με μια απλή εντολή, η οποία δημιουργεί τον κατάλογο εργασίας του site και μια σειρά από αρχεία και υποκαταλόγους για το κτίσιμό του.

Το site μας χρειάζεται κι ένα theme. Τα themes του Hugo καθορίζουν το **πώς φαίνεται ένα Hugo site** στους επισκέπτες του, βρίσκονται δε όλα οργανωμένα στη σελίδα Hugo Themes (https://themes.gohugo.io). Είναι εύκολο να εντοπίσουμε ένα ή περισσότερα themes που να ταιριάζουν με ό,τι έχουμε κατά νου για το site μας, αφού εκτός από την ποικιλία παρέχεται και δυνατότητα preview (βλ. κουμπί **Demo**). Υπόψιν επίσης ότι μπορούμε, αν το επιθυμούμε, να τροποποιούμε τα themes. Για τις ανάγκες της παρουσίασής μας ας εστιάσουμε την προσοχή μας στο μινιμαλιστικό και σκοτει-

νό (κυριολεκτικά) theme ονόματι After Dark (https://themes.gohugo.io/after-dark). Μεταβαίνουμε στον κατάλογο themes του site μας (π.χ., cd ~/colder-dot-xyz/themes) και κλωνοποιούμε το theme από τη σελίδα του στο GitHub:

```
mbpr15:themes cvar$ git clone \
> https://github.com/comfusion/after-dark.git
```

```
[mbpr15:~ cvar$
[mbpr15:~ cvar$ cd ~/colder-dot-xyz/themes
[mbpr15:themes cvar$
[mbpr15:themes cvar$ git clone https://github.com/comfusion/after-dark.git
 Cloning into 'after-dark'...
 remote: Counting objects: 2211, done.
 remote: Total 2211 (delta 0), reused 0 (delta 0), pack-reused 2211
 Receiving objects: 100% (2211/2211), 17.21 MiB | 2.28 MiB/s, done.
 Resolving deltas: 100% (1212/1212), done.
[mbpr15:themes cvar$
[mbpr15:themes cvar$ ls -lh
 total 0
 drwxr-xr-x  15 cvar  staff   510B 17 des 10:28 after-dark
[mbpr15:themes cvar$
[mbpr15:themes cvar$ ls -lh after-dark
 total 200
 -rw-r--r--  1 cvar  staff    12K 17 des 10:28 CHANGELOG.md
 -rw-r--r--  1 cvar  staff   500B 17 des 10:28 COPYING
 -rw-r--r--  1 cvar  staff    30K 17 des 10:28 README.md
 drwxr-xr-x  4 cvar  staff   136B 17 des 10:28 archetypes
 drwxr-xr-x  3 cvar  staff   102B 17 des 10:28 bin
 drwxr-xr-x  6 cvar  staff   204B 17 des 10:28 images
 drwxr-xr-x  8 cvar  staff   272B 17 des 10:28 layouts
 -rw-r--r--  1 cvar  staff   1,1K 17 des 10:28 package.json
 drwxr-xr-x  4 cvar  staff   136B 17 des 10:28 static
 -rw-r--r--  1 cvar  staff   944B 17 des 10:28 theme.toml
 -rw-r--r--  1 cvar  staff    38K 17 des 10:28 yarn.lock
 mbpr15:themes cvar$
```

Λήψη του λιτού, απέριττου και γοητευτικά σκοτεινού Hugo theme ονόματι After Dark, απευθείας από το GitHub.

Είμαστε σχεδόν έτοιμοι για την προσθήκη περιεχομένου στο ολοκαίνουργιο static site μας.

Βασικές ρυθμίσεις

Πριν αφήσουμε τη συγγραφική μας δημιουργικότητα ελεύθερη, οφείλουμε να τακτοποιήσουμε κάποιες απαραίτητες για το site ρυθμίσεις. Με τον text editor της προτίμησής μας ανοίγουμε το αρχείο config.toml, φυσικά εντός του καταλόγου ~/colder-dot-xyz. Περιλαμβάνει τρεις μόλις γραμμές, τις οποίες εμείς τροποποιήσαμε ως ακολούθως:

```
baseURL = "https://colder.xyz/"
languageCode = "el-gr"
title = "Colder is Better"
```

Βεβαίως, στη μεταβλητή baseURL εσείς θα αναθέσετε την ονομαστική ή την αριθμητική διεύθυνση του site στο οποίο θα καταλήγει το περιεχόμενο που παράγετε. Επίσης, αναλόγως θα τροποποιήσετε και τον τίτλο του εν λόγω δικτυακού τόπου. Μετά από τις τρεις αυτές γραμμές προσθέσαμε κι άλλες δύο:

```
theme = "after-dark"
pygmentsStyle = "native"
```

Στη μεταβλητή theme βάλτε το όνομα του καταλόγου που βρίσκεται εντός του ~/colder-dot-xyz/themes κι αντιστοιχεί στο theme που θέλετε για το static site σας. Όσο για τις τιμές που δέχεται η μεταβλητή pygmentsStyle, δείτε τη σελίδα με το Pygments online demo (http://pygments.org/demo).

Υπάρχουν κι άλλες ενδιαφέρουσες ρυθμίσεις που χρειάζεται να γίνουν μέσω του αρχείου config.toml – και σε λίγο θα φροντίσουμε σχετικά. Προς το παρόν όμως είναι καλή ιδέα να δημιουργήσουμε το πρώτο μας post και να δούμε, τοπικά, πώς θα φαίνεται στους επισκέπτες του site μας.

Δημιουργία σελίδας About

Το περιεχόμενο που παράγουμε για το Hugo εξ ορισμού γράφεται σε Markdown. Τα σχετικά αρχεία, με κατάληξη .md, αποθηκεύονται είτε μέσα στον κατάλογο content είτε μέσα σε κάποιον υποκατάλογό του. Κάθε αρχείο του είδους συμβάλλει στη δημιουργία μιας νέας δημοσίευσης ή αλλιώς άρθρου. Προσέξτε τώρα: Μια οποιαδήποτε νέα δημοσίευση, η οποία αντιστοιχεί σε αρχείο .md κάτω από τον κατάλογο content, δεν τη φτιάχνουμε **απευθείας** μέσω ενός text editor. Αντίθετα, βεβαιωνόμαστε ότι είμαστε **εντός του καταλόγου του site** (π.χ., cd ~/colder-dot-xyz) και στο τερματικό μας πληκτρολογούμε κάτι σαν hugo new post-name.md. Κατ' αυτόν τον τρόπο, στην αρχή του post-name.md περιλαμβάνονται **editable metadata** για την υπό προετοιμασία δημοσίευση. Όλες οι σχετικές πληροφορίες αποτελούν μια ειδική ενότητα που στον κόσμο του Hugo ονομάζεται **front matter**. Αλλά ας τα δούμε όλα αυτά μέσα από ένα πολύ συγκεκριμένο παράδειγμα: τη δημιουργία μιας σελίδας About, η οποία, θα συμφωνήσετε κι εσείς, είναι μάλλον απαραίτητη για κάθε site που σέβεται τον εαυτό του.

Πού πρέπει να βρίσκεται το αντίστοιχο αρχείο .md; Μπορούμε να το ονομάσουμε όπως θέλουμε; Οι απαντήσεις διαφέρουν από theme σε theme, όπως σωστά θα υποθέσατε. Για το After Dark, το αρχείο της σελίδας About πρέπει να ονομάζεται about.md και να είναι αποθηκευμένο **στη ρίζα** του καταλόγου content. Πληκτρολογούμε, λοιπόν…

```
mbpr15:colder-dot-xyz cvar$ hugo new about.md
```

...και το αρχείο about.md ακαριαία δημιουργείται εντός του content. Ανοίγοντάς το με τον text editor της προτίμησής μας, πάνω πάνω βλέπουμε το front matter:

```
---
title: "About"
date: 2017-12-21T07:06:46+01:00
draft: true
---
```

Για το παράδειγμά μας δεν έχει και πολύ νόημα να αλλάξουμε τις τιμές των μεταβλητών αριστερά του χαρακτήρα :, ενδεχομένως ούτε και να προσθέσουμε νέες μεταβλητές. Έχετε πάντως υπόψη ότι το προκαθορισμένο περιεχόμενο του front matter εξαρτάται από το εκάστοτε theme. Σε κάθε περίπτωση, αξίζει να έχουμε από τώρα μιαν εικόνα για τις μεταβλητές που είναι πιθανό να δούμε ή/και να προσθέσουμε στο front matter ενός νέου post.

- **date**: Ημερομηνία και ώρα δημιουργίας του post. Αν το επιθυμούμε αλλάζουμε τόσο την ημερομηνία, όσο και την ώρα.
- **description**: Μια περιγραφή του περιεχομένου. Παραδείγματα: "Περί αυτού του blog και γιατί θα θέλατε να νοιαστείτε", "Πώς θα φτιάξετε ένα οικιακό Ceph cluster χωρίς να ξοδέψετε μια περιουσία" κ.ο.κ.
- **draft**: Αν η τιμή αυτής της μεταβλητής είναι true, τότε δίνοντας την εντολή hugo, για το τοπικό κτίσιμο του site, το rendering του συγκεκριμένου post παρακάμπτεται. Αν όμως στο hugo δώσουμε την οδηγία --buildDrafts, τότε το rendering της σελίδας **δεν** παρακάμπτεται.
- **slug**: Καθορισμός προσαρμοσμένου URL για τη σελίδα (του post).
- **title**: Ένας τίτλος για τη δημοσίευση.
- **tags**: Ένα ή περισσότερα tags, μέσα σε αγκύλες, για το post. Κάθε tag περικλείεται από διπλά εισαγωγικά και μεταξύ τους χωρίζονται με κόμματα. Παράδειγμα: tags: ["about", "cold", "winter"].
- **categories**: Μία ή περισσότερες κατηγορίες, μέσα σε αγκύλες, στις οποίες θέλουμε να ανήκει το post. Κάθε κατηγορία περικλείεται από διπλά εισαγωγικά και μεταξύ τους χωρίζονται με κόμματα.

Περισσότερα για τις μεταβλητές του front matter μπορείτε να διαβάσετε στο επίσημο documentation (https://gohugo.io/content-management/front-matter/#front-matter-variables).

Και τώρα που έχουμε μια καλή ιδέα για το front matter και τη χρησιμότητά του, ίσως θελήσουμε να προσθέσουμε μία-δύο μεταβλητές κι από κάτω να γράψουμε το περιεχόμενό μας – σε Markdown, υπενθυμίζουμε. Αν επιθυμούμε να βλέπουμε **ζωντανά** πώς θα δείχνει η σελίδα στο site καθώς γράφουμε, διορθώνουμε κι αποθηκεύουμε, τότε σε ένα τερματικό μεταβαίνουμε στον κατάλογο του site (π.χ., cd ~/colder-dot-xyz) και πληκτρολογούμε hugo server --buildDrafts --watch. Για την οδηγία --buildDrafts (σύντομη εκδοχή -D) μιλήσαμε ήδη, ενώ η λειτουργία της οδηγίας --watch (σύντομη εκδοχή -w) είναι, νομίζουμε, προφανής.

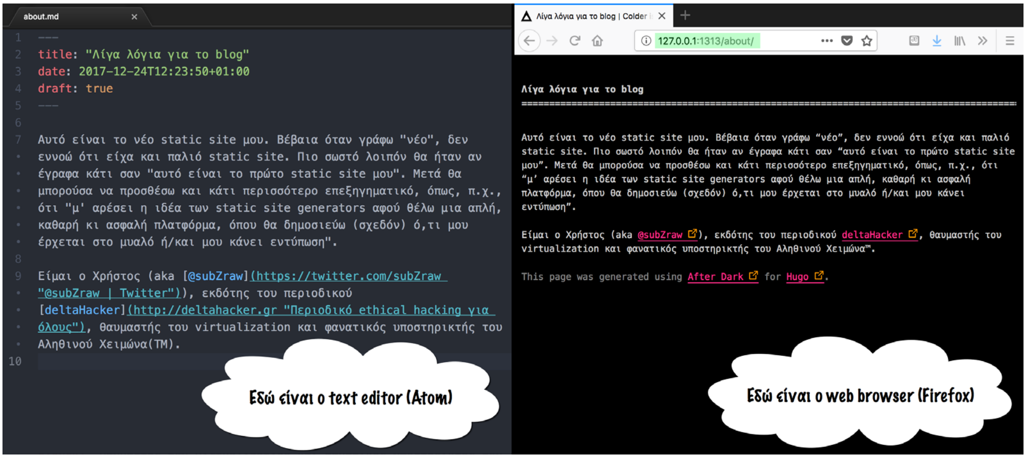
```
colder-dot-xyz — sub0@box: ~/lists — hugo server --buildDrafts --watch — 115×30 — ⌘1
mbpr15:colder-dot-xyz cvar$
mbpr15:colder-dot-xyz cvar$ hugo server --buildDrafts --watch
Started building sites ...

Built site for language en:
1 of 1 draft rendered
0 future content
0 expired content
1 regular pages created
6 other pages created
0 non-page files copied
0 paginator pages created
0 tags created
0 categories created
total in 6 ms
Watching for changes in /Users/cvar/colder-dot-xyz/{data,content,layouts,themes,static}
Serving pages from memory
Running in Fast Render Mode. For full rebuilds on change: hugo server --disableFastRender
Web Server is available at http://localhost:1313/ (bind address 127.0.0.1)
Press Ctrl+C to stop
```

Ζωντανό (και ταχύτατο) rendering του Hugo site μας, τοπικά. Λόγω της οδηγίας --buildDrafts, τα posts που είναι υπό ανάπτυξη δεν εξαιρούνται από το rendering. Έτσι, σε συνδυασμό και με την οδηγία --watch, καθώς γράφουμε ένα post σε Markdown μπορούμε και βλέπουμε πώς ακριβώς θα φαίνεται στους επισκέπτες του site μας.

Το rendered site το βλέπουμε τοπικά ανοίγοντας έναν web browser και μεταβαίνοντας στη διεύθυνση http://127.0.0.1:1313. Για το theme που έχουμε επιλέξει, προκειμένου να εμφανίσουμε στον web browser τη σελίδα About οφείλουμε να επισκεφτούμε τη διεύθυνση http://127.0.0.1:1313/about. Δείτε το screenshot που ακολουθεί, διαβάστε και την αντίστοιχη περιγραφή.

Αφού ολοκληρώσουμε τη συγγραφή της σελίδας About, δεν χρειαζόμαστε άλλο το τοπικό rendering. Πηγαίνουμε στο τερματικό όπου το είχαμε ξεκινήσει (γράφοντας, π.χ., hugo server --buildDrafts --watch) κι απλά πατάμε το συνδυασμό πλήκτρων [CTRL+C]. Φυσικά, πριν ανεβάσουμε το περιεχόμενο στο site μας, καλό θα ήταν να ετοιμάσουμε κι ένα κάποιο περισσότερο ενδιαφέρον post. Επιπρόσθετα, χρειάζονται και μερικές επεμβάσεις στο theme.



Συγγραφή post σε Markdown (αριστερά) και ταυτόχρονη επισκόπηση της σελίδας (δεξιά), όπως θα τη βλέπουν οι επισκέπτες του Hugo static site μας.

Προσαρμογή theme

Από τη στιγμή που θα επιλέξουμε ένα theme (https://themes.gohugo.io) για το Hugo static site μας, το πρώτο πράγμα που οφείλουμε να κάνουμε είναι να διαβάσουμε για το theme και συγκεκριμένα για το πώς το προσαρμόζουμε στις ανάγκες μας. Ψάχνοντας για το τι μπορούμε να κάνουμε (https://themes.gohugo.io/after-dark) με το After Dark, τροποποιήσαμε ελαφρώς το αρχείο config.toml ώστε μεταξύ άλλων να ορίσουμε ένα απλό και βολικό μενού για τον ευκολότερο εντοπισμό των άρθρων στο site. Δείτε:

```
baseURL = "https://colder.xyz/"
title = "Colder is Better"
description = "Seemingly Random Postings"
languageCode = "el-GR"

pygmentsStyle = "native"
theme = "after-dark"
canonifyurls = true
paginate = 5

enableRobotsTXT = true

SectionPagesMenu = "main"

[params]
```

```
        powered_by = false
        author = "Christos Varelas"
        show_menu = true

    [menu]
        [[menu.main]]
            name = "Home"
            weight = 1
            identifier = "home"
            url = "/"
        [[menu.main]]
            name = "Posts"
            weight = 2
            identifier = "post"
            url = "/post/"
        [[menu.main]]
            name = "About"
            weight = 3
            identifier = "about"
            url = "/about"
```

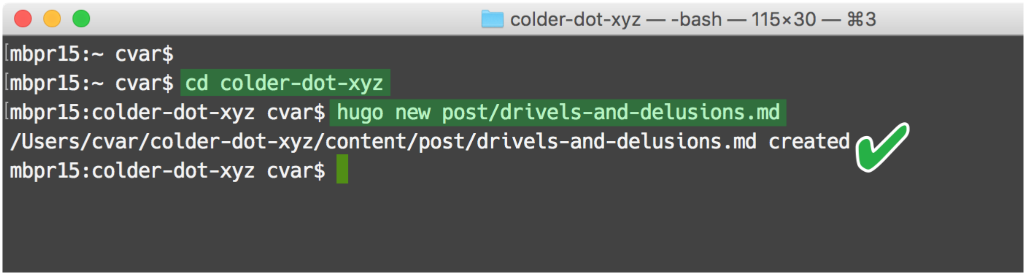
Άσχετα αν θα χρησιμοποιήσετε κι εσείς το After Dark ή όχι, το ηθικό δίδαγμα εδώ είναι ότι τροποποιούμε την όψη και τη συμπεριφορά του theme επεμβαίνοντας στο αρχείο config.toml, εντός του καταλόγου του site. Υπάρχουν κι άλλοι τρόποι για την αλλαγή ενός theme, ξεφεύγουν όμως από το σκοπό του παρόντος άρθρου.

Δημιουργία post

Γνωρίζουμε, πλέον, ότι αρχικά πρέπει να πάμε στον κατάλογο του site μας (cd ~/colder-dot-xyz) κι αμέσως μετά να δημιουργήσουμε το αρχείο .md του νέου post, κάτω από τον κατάλογο content/post: απλά πληκτρολογούμε hugo new post/drivels-and-delusions.md. Για τις ανάγκες της παρουσίασής μας θα αναδημοσιεύσουμε στο static site το editorial (https://deltahacker.gr/deltahacker060editorial) από το deltaHacker 060 (https://deltahacker.gr/deltahacker060).

Ανοίγουμε το αρχείο content/post/drivels-and-delusions.md με κάποιον text editor και εξετάζουμε λίγο το front matter, το οποίο στην περίπτωσή μας έχει ως ακολούθως:

```
    ---
    title: "Drivels and Delusions"
```



Τα πρώτα απλά βήματα για τη δημιουργία ενός νέου post για το Hugo static site μας. Για τις ανάγκες της παρουσίασης αναδημοσιεύουμε το editorial του deltaHacker 060.

```
date: 2017-12-26T10:20:00+01:00
draft: true
---
```

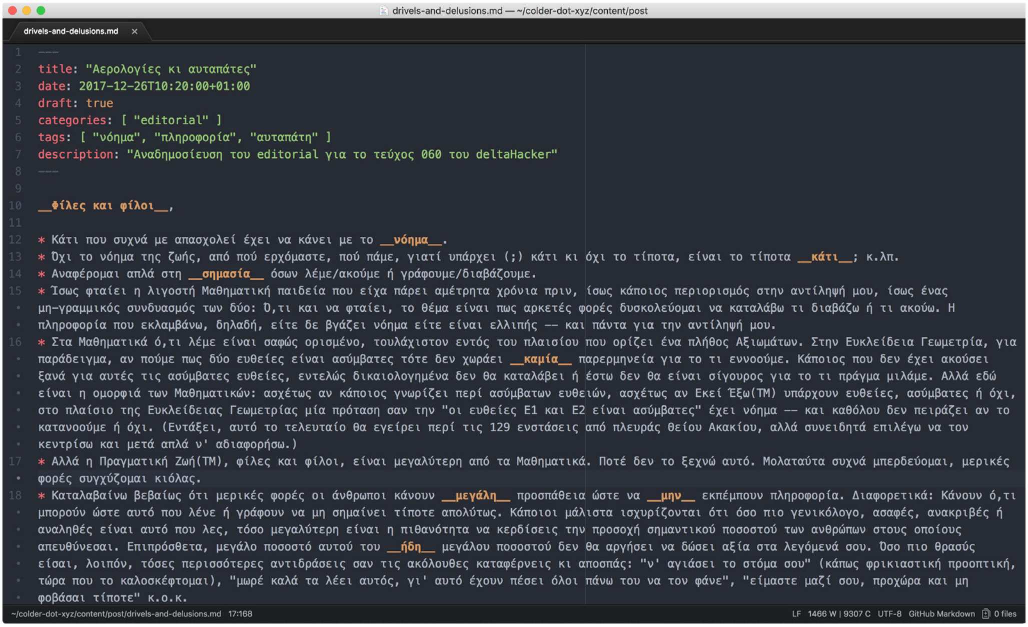
Παρατηρήστε ότι ο τίτλος παράγεται αυτόματα, από το όνομα του αρχείου .md (οι παύλες αντικαθίστανται από κενά). Όπως έχουμε ήδη αναφέρει, το front matter μπορούμε να το τροποποιούμε ή/και να το επαυξάνουμε. Δείτε, για παράδειγμα, το front matter για το editorial που ετοιμαζόμαστε να δημοσιεύσουμε:

```
---
title: "Αερολογίες κι αυταπάτες"
date: 2017-12-26T10:20:00+01:00
draft: true
categories: [ "editorial" ]
tags: [ "νόημα", "πληροφορία", "αυταπάτη" ]
description: "Αναδημοσίευση του editorial για το τεύχος 060 του
deltaHacker"
---
```

Νομίζουμε ότι αυτή η νεότερη εκδοχή είναι πολύ περισσότερο αντιπροσωπευτική του περιεχομένου που πρόκειται ν' ακολουθήσει. Παρεμπιπτόντως, δεν υπάρχει λόγος να το καθυστερούμε άλλο: είναι ώρα να το προσθέσουμε, σε μορφή Markdown, κάτω από το front matter.

Μεταφορά περιεχομένου στον web server

Αφού τελειώσουμε με τη συγγραφή του νέου post και είμαστε ικανοποιημένοι με το πώς θα δείχνει στους επισκέπτες του site μας, πριν εγκαταλείψουμε τον editor είναι καλή ιδέα να αλλάξουμε την τιμή της μεταβλητής draft από true σε false. Αμέσως μετά, σταματάμε αν χρειαστεί το live preview (στην κονσόλα που είχαμε δώσει κάτι σαν hugo server -D -w τώρα πατάμε [CTRL+C]) κι από τον κατάλογο του site δημιουργούμε **όλο** το περιεχόμενο του site πληκτρολογώντας, απλά, hugo. Προσοχή: Αν



Ο Atom είναι ο αγαπημένος μας text editor για το περιβάλλον γραφικών.

σε κάποια άρθρα έχουμε αφήσει τη μεταβλητή draft να ισούται με true και **θέλουμε** αυτά τα άρθρα να προετοιμαστούν για δημοσίευση στο site, τότε αντί για hugo δίνουμε hugo --buildDrafts ή, απλούστερα, hugo -D. Η εντολή hugo δημιουργεί το περιεχόμενο και το αποθηκεύει τοπικά, στον κατάλογο public. Ό,τι περιλαμβάνει ο συγκεκριμένος κατάλογος αρκεί να το στείλουμε στον (απομακρυσμένο) κατάλογο από τον οποίο ο web server μας σερβίρει περιεχόμενο. Ένα καλό εργαλείο για τη δουλειά αυτή είναι το rsync. Για παράδειγμα, ο δικός μας web server βρίσκεται στο μηχάνημα box.colder.xyz και το περιεχόμενο που σερβίρει το διαβάζει από τον κατάλογο /home/user-data/www/default. Εκεί έχουμε έναν χρήστη, τον sub0, με δικαιώματα εγγραφής εντός του καταλόγου default. Επίσης, στο απομακρυσμένο μηχάνημα τρέχει η υπηρεσία OpenSSH. Από τον τοπικό κατάλογο ~/colder-dot-xyz, λοιπόν, συγχρονίζουμε το περιεχόμενο γράφοντας rsync -av public/* sub0@box.colder.xyz:/home/user-data/www/default/. Στον κατάλογο public είναι πιθανό να βρίσκονται drafts από άρθρα που πρόκειται να δημοσιεύσουμε σε μελλοντικό χρόνο ή/και αρχεία από άλλα themes που έχουμε δοκιμάσει. Γενικά, μιλάμε για υλικό που δεν θέλουμε να καταλήγει στον web server μας. Αν υποψιαζόμαστε ότι τέτοια αρχεία πράγματι υπάρχουν στον public, τότε πριν τρέξουμε το hugo, για το κτίσιμο του περιεχομένου τοπικά, είναι καλή ιδέα να διαγράψουμε εντελώς τον κατάλογο public. Μετά θα δώσουμε hugo κι ο κατάλογος public με το φρέσκο περιεχόμενο θα δημιουργηθεί εκ νέου κι ακαριαία (μην ξεχνάτε ότι η ταχύτητα είναι ένα από τα βασικά γνωρίσματα του Hugo). Τέλος, θα συγχρονίσουμε και το περιεχόμενο του public με το rsync (ή με όποιον άλλον τρόπο προτιμάμε, τέλος πάντων).

Σημείωση. Δουλεύετε σε περιβάλλον Mac OS κι έχετε tags ή κατηγορίες στα ελληνικά; Αφού ανεβάσετε το περιεχόμενο του public στον κατάλογο του απομακρυσμένου web server, θα διαπιστώσετε ότι κάποια URLs δεν θα δουλεύουν σωστά (π.χ., εκείνα που εμπεριέχουν ένα tag στα ελληνικά). Η λύση είναι να μη χρησιμοποιήσετε την προεγκατεστημένη έκδοση του rsync (σειρά 2.6.x) αλλά κάποια νεότερη (της σειράς 3.1.x). Ένας τρόπος για να έχετε στο Mac OS νεότερη έκδοση του rsync (και πολλών-πολλών άλλων, Unix-like tools), είναι μέσω του Homebrew (https://brew.sh). Με το νέο, λοιπόν, rsync, θα συγχρονίζετε το περιεχόμενο του public γράφοντας κάτι τέτοιο:

```
mbpr15:colder-dot-xyz cvar$ rsync -av \
> --iconv=utf-8-mac,utf-8 public/* \
> sub0@box.colder.xyz:/home/user-data/www/default/
```

Favicon για το static site;

Με το After Dark, το theme που επιλέξαμε για το παρόν άρθρο, σε κάθε σελίδα έχουμε κι ένα απλό favicon σε μορφή SVG (είναι ένα μαύρο, κεφαλαίο δέλτα). Αν θελήσετε να το αλλάξετε, τότε ξεκινήστε φτιάχνοντας τον κατάλογο partials/head εντός του layouts (και πάντα μέσα στον τοπικό κατάλογο του site – εν προκειμένω στον ~/colder-dit-xyz):

```
mbpr15:colder-dot-xyz cvar$ mkdir -p layouts/partials/head
```

Τώρα, κάτω από τον κατάλογο head δημιουργήστε ένα νέο αρχείο με όνομα favicon.html και περιεχόμενο που θα μοιάζει με το ακόλουθο:

```
<link rel="icon" sizes="any" href="https://www.dropbox.com/.../
fn.ico?dl=1">
```

Το δικό σας link προς το favicon θα είναι, φυσικά, διαφορετικό. Αυτό που εμείς κάναμε είναι να κατεβάσουμε ένα ωραίο favicon από το freefavicon (https://www.freefavicon.com/freefavicons) και να το ανεβάσουμε στο Dropbox μας. Μετά δημιουργήσαμε ένα public link προς το αρχείο .ico του favicon στο Dropbox και, τέλος, το σχετικό URL το βάλαμε στο favicon.html.

Χαρείτε **πάρα πολύ** και χωρίς την παραμικρή συστολή το ολοκαίνουργιο static site σας, με το Hugo. Επιλέξτε ένα όμορφο theme, πειράξτε το λίγο κι από εκεί και πέρα επικεντρωθείτε στην παραγωγή περιεχομένου και μόνο.

Skill: Beginner
Tags: Firefox, openSUSE, Quantum

Εγκαταστήστε τον καινούργιο Firefox στο openSUSE Leap

Ο Firefox 57 είναι γρήγορος. Η απλή κι απέριττη αυτή πρόταση είναι 110% αληθής. Πραγματικά δεν χρειάζεται να γράψουμε τίποτε άλλο για να κερδίσουμε την προσοχή σας, και έτσι να διαπιστώσετε κι από μόνοι σας πόσο τρομακτικά γρήγορος είναι.

Άλλωστε το πιθανότερο είναι ότι ήδη χρησιμοποιείτε τον νέο browser, οπότε ξέρετε από πρώτο χέρι για την ασύλληπτη ταχύτητά του. Αν όμως έχετε μηχάνημα με το υπέροχο openSUSE 42.3, θα διαπιστώσετε ότι ο Firefox 57 δεν υπάρχει στα αποθετήρια της διανομής. Αναλυτικότερα, η νεότερη έκδοση που περιλαμβάνεται στο OSS repository του Leap 42.3 είναι η 52.5.



Ο ταχύτατος Firefox 57 δεν υπάρχει στο επίσημο repository με τα updates του openSUSE Leap 42.3. Μπορούμε μολαταύτα να τον έχουμε, αρκεί να προσθέσουμε στο σύστημα το κατάλληλο αποθετήριο λογισμικού.

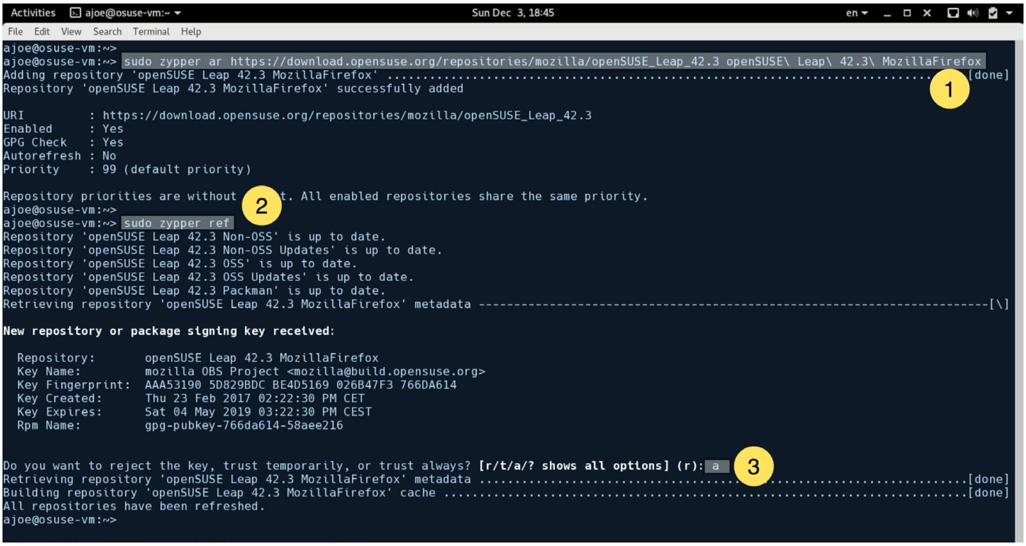
Προκειμένου να έχουμε την πλέον πρόσφατη έκδοση του Firefox **χωρίς** μεταγλωττίσεις ή χειροκίνητες εγκαταστάσεις binaries, αρκεί να προσθέσουμε στο σύστημά μας το κατάλληλο software repository. Το URL του ζητούμενου αποθετηρίου είναι στο https://download.opensuse.org/repositories/mozilla/openSUSE_Leap_42.3. Για να το προσθέσουμε, σε ένα τερματικό πληκτρολογούμε...

```
sudo zypper ar \
> https://download.opensuse.org/repositories/mozilla/openSUSE_Leap_42.3 \
> openSUSE\ Leap\ 42.3\ MozillaFirefox
```

...όπου βεβαίως δεν είναι υποχρεωτικό να επιλέξετε κι εσείς το όνομα "openSUSE Leap 42.3 MozillaFirefox" για το νέο repository. Στη συνέχεια φρεσκάρουμε τα περιεχόμενα όλων των τοπικών αποθετηρίων, γράφοντας:

```
sudo zypper ref
```

Το αποθετήριο openSUSE Leap 42.3 MozillaFirefox είναι νέο για το σύστημά μας, οπότε το zypper θα ζητήσει οδηγίες για το τι πρέπει να κάνει με το κλειδί που έχει χρησιμοποιηθεί για την υπογραφή των πακέτων που περιέχει. Οι επιλογές μας είναι: απόρριψη του κλειδιού (reject, r), προσωρινή αποδοχή (trust temporarily, t) ή μόνιμη αποδοχή (trust always, a). Το συγκεκριμένο repository δεν έχουμε λόγο ώστε να **μην** το θεωρούμε έμπιστο, οπότε επιλέγουμε τη μόνιμη αποδοχή.

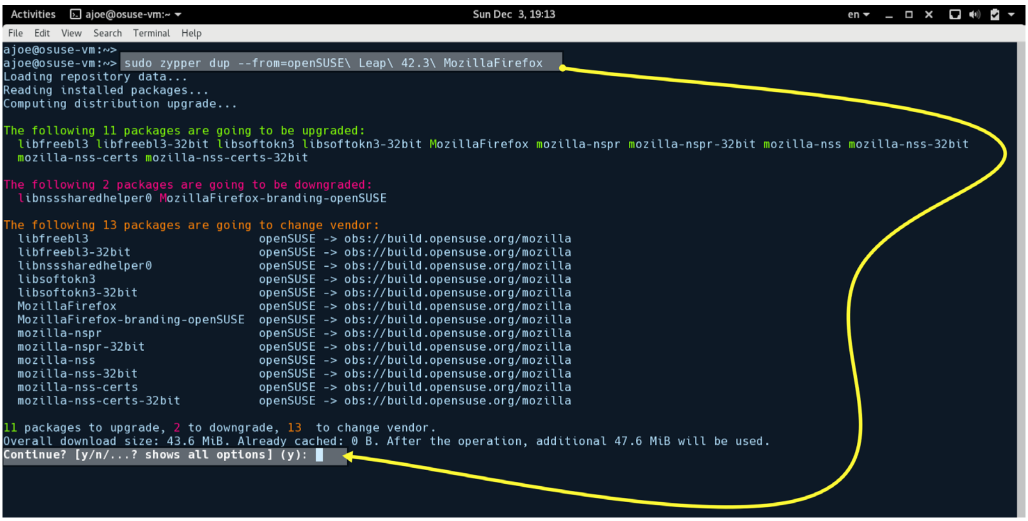


Σε τρία βήματα: Προσθήκη software repository με τον πλέον πρόσφατο Firefox, στο openSUSE Leap 42.3.

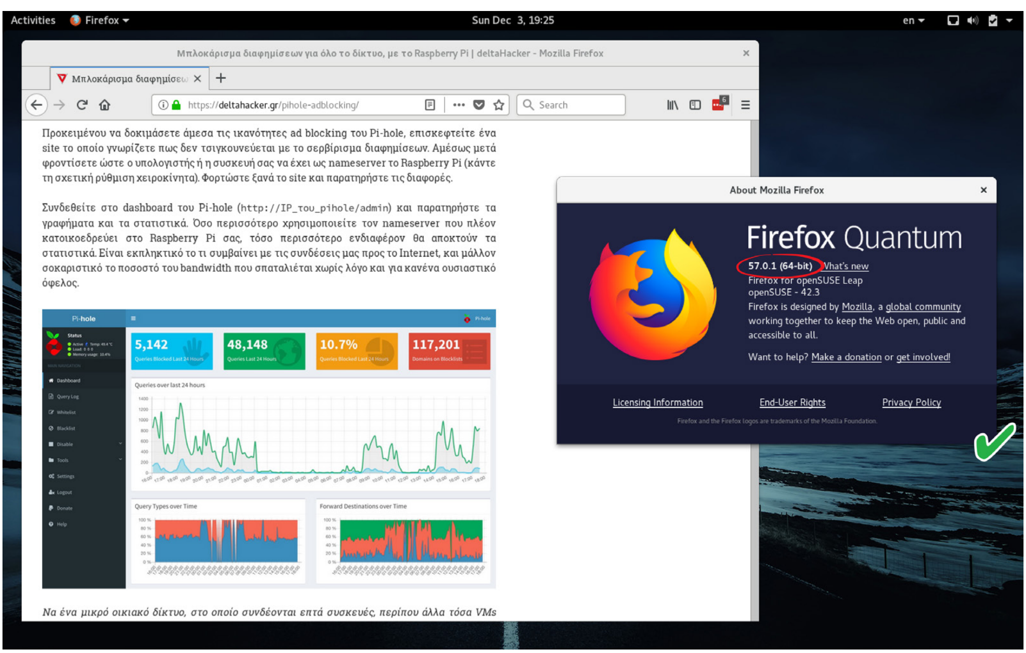
Τώρα, για να εγκαταστήσουμε την τρέχουσα έκδοση του Firefox λαμβάνοντας υπόψη το **νέο** repository, στο τερματικό μας δίνουμε κάτι σαν το ακόλουθο:

```
sudo zypper dup --from=openSUSE\ Leap\ 42.3\ MozillaFirefox
```

(στην ερώτηση του zypper για το αν πρέπει να προχωρήσει, απαντάμε βεβαίως καταφατικά). Το ίδιο θα κάνουμε και κάθε φορά που θα θέλουμε να αναβαθμίσουμε τον Firefox – εννοείται με βάση το repository που μόλις προσθέσαμε.



Αναβάθμιση με βάση το νέο repository για τον Firefox, ώστε να έχουμε πάντοτε την πλέον πρόσφατη έκδοση του browser.



Επιτέλους, ο ταχύτατος Firefox 57.0.1, με τη μηχανή Quantum, στο αγαπημένο openSUSE Leap 42.3!

Skill: Advanced
Tags: KVM, QEMU, libvirt, switch, bridge, NAT, dnsmasq, DHCP, resolving, port forwarding

Διαχείριση εικονικών τοπικών δικτύων στον κόσμο του libvirt

Στο τεύχος 060 του deltaHacker δείξαμε μεταξύ άλλων πώς φτιάχνουμε και κλωνοποιούμε QEMU/KVM VMs, εργαζόμενοι αποκλειστικά από τη γραμμή εντολών. Είναι ώρα να ασχοληθούμε και με τα εικονικά τοπικά δίκτυα, παραμένοντας βεβαίως στην αγαπημένη μας text console.

Γιατί όμως να προτιμήσουμε την κονσόλα, όταν διαθέτουμε εύχρηστες εφαρμογές για το περιβάλλον γραφικών, όπως, π.χ., είναι ο Virtual Machine Manager; Όπως φανταζόσαστε, υπάρχουν περισσότεροι από ένας λόγοι. Ίσως ο VM host να είναι ένα απομακρυσμένο μηχάνημα, χωρίς GUI. Ίσως πάλι σχεδιάζουμε να γράψουμε μερικά scripts για τη γρήγορη δημιουργία εικονικών εργαστηρίων για development ή για τις ανάγκες μιας παρουσίασης. Είναι επίσης πιθανό να επιδιώκουμε την αξιολόγηση ενός cluster, συγκεκριμένων υπηρεσιών ή μιας νέας τεχνολογίας **πριν** από το deployment σε πραγματικό hardware. Ό,τι κι αν ισχύει, η εργασία από την text console έχει μοναδικά πλεονεκτήματα τα οποία απουσιάζουν από το όποιο GUI.

Ξεκίνημα με απαραίτητη παρένθεση

Πριν περάσουμε στο προκείμενο αξίζει να αναφερθούμε εν τάχει σε συγκεκριμένους όρους, τους οποίους πολύ συχνά χρησιμοποιούμε χωρίς όμως να έχουμε ορίσει επακριβώς. Προς αποφυγή παρερμηνειών, λοιπόν, σκεφτόμαστε ότι καλό θα ήταν να συζητήσουμε λίγο περί **KVM**, **QEMU** και **libvirt**.

- Το **KVM** (Kernel-based Virtual Machine) είναι μια τεχνολογία που επιτρέπει στον πυρήνα του Linux να λειτουργεί ως hypervizor, μ' άλλα λόγια να είναι ικανός για τη φιλοξενία εικονικών μηχανών. Από την έκδοση 2.6.20 του πυρήνα, το KVM αποτελεί υποσύστημά του. Διατίθεται επίσης ως loadable module για τα FreeBSD και illumos (https://wiki.illumos.org/display/illumos/illumos+Home). Το KVM απαιτεί από τον φυσικό επεξεργαστή να παρέχει τα λεγόμενα hardware *virtualization extensions*. Αν και αρχικά υποστήριζε μόνο x86/x86-64 CPUs (πάντα με virtualization extensions), σταδιακά άρχισε να υποστηρίζει κι άλλες αρχιτεκτονικές επεξεργαστών, όπως, π.χ., PowerPC και ARM. Οι εικονικές μηχανές που βασίζονται στο KVM μπορούν να τρέχουν μια ευρεία γκάμα λειτουργικών συστημάτων, με τα Windows, Linux, BSD και OS X να αποτελούν τους συνήθεις

υπόπτους. Αξίζει να υπογραμμίσουμε ότι το KVM από μόνο του **δεν** υλοποιεί κάποιου είδους εξομοίωση. Αντίθετα, σε εφαρμογές του userspace παρέχει το ειδικό αρχείο συσκευής /dev/kvm, ώστε αυτές να είναι σε θέση να δημιουργούν εικονικές μηχανές και, το σημαντικότερο, να τις τρέχουν **αποδοτικά**.

- Το **QEMU** (Quick EMUlator) είναι μια εφαρμογή δημιουργίας και διαχείρισης εικονικών μηχανών, η οποία κάνει χρήση του /dev/kvm **χωρίς** όμως να το απαιτεί. Αναλυτικότερα, το QEMU είναι ικανό να εξομοιώνει πλήρως το hardware ενός υπολογιστή κι αυτός με τη σειρά του είναι ικανός να τρέχει μια σειρά λειτουργικών συστημάτων. Όταν το QEMU εντοπίζει το /dev/kvm, τότε πρόθυμα το χρησιμοποιεί και οι σχετικές εικονικές μηχανές τρέχουν με ταχύτητες **παραπλήσιες** του φυσικού υπολογιστή που τις φιλοξενεί. Κάθε φορά, λοιπόν, που γράφουμε **QEMU/KVM**, ουσιαστικά αναφερόμαστε σε εικονικές μηχανές QEMU οι οποίες φιλοξενούνται σε υπολογιστή που υποστηρίζει την τεχνολογία του KVM.
- Το δε **libvirt** είναι κάτι σαν εργαλειοθήκη για τη διαχείριση διαφορετικών virtualization platforms. Παρέχει ένα API σε C και bindings για γλώσσες όπως Python, Ruby και JavaScript. Επίσης, μία υπηρεσία (daemon) καθώς κι εργαλεία διαχείρισης. Χάρη στο libvirt οι προγραμματιστές είναι σε θέση ν' αναπτύσσουν εφαρμογές **χωρίς** να νοιάζονται για το ποιος είναι ο hypervizor στο παρασκήνιο. Επειδή το libvirt ξέρει πώς να μιλά σε QEMU/KVM, VMware, VirtualBox, bhyve κι άλλες πλατφόρμες, μια εφαρμογή που χρησιμοποιεί το libvirt API είναι σε θέση να λέει *ξεκίνα την τάδε μηχανή, δώσε μου όλες τις ενεργές μηχανές, διπλασίασε τη μνήμη αυτής της μηχανής παραπέρα, πάρε ένα snapshot, πρόσθεσε έναν δίσκο κ.ο.κ.*, κι από εκεί και πέρα να μη νοιάζεται για τις λεπτομέρειες της υλοποίησης. Και δεν νοιάζεται διότι κάθε φορά που τα λέει με το libvirt API, είναι μετά ευθύνη του τελευταίου να πραγματοποιήσει τις όποιες επιθυμίες της — ή τουλάχιστον να προσπαθήσει κι αν αποτύχει να αναφέρει το λόγο. Μερικά παραδείγματα εφαρμογών που χρησιμοποιούν το libvirt API είναι ο Virtual Machine Manager (https://deltahacker.gr/?p=15425), το virsh (συμπεριλαμβάνεται στο βασικό πακέτο του libvirt, https://deltahacker.gr/?p=17646), καθώς και το virt-clone (https://deltahacker.gr/?p=17662).

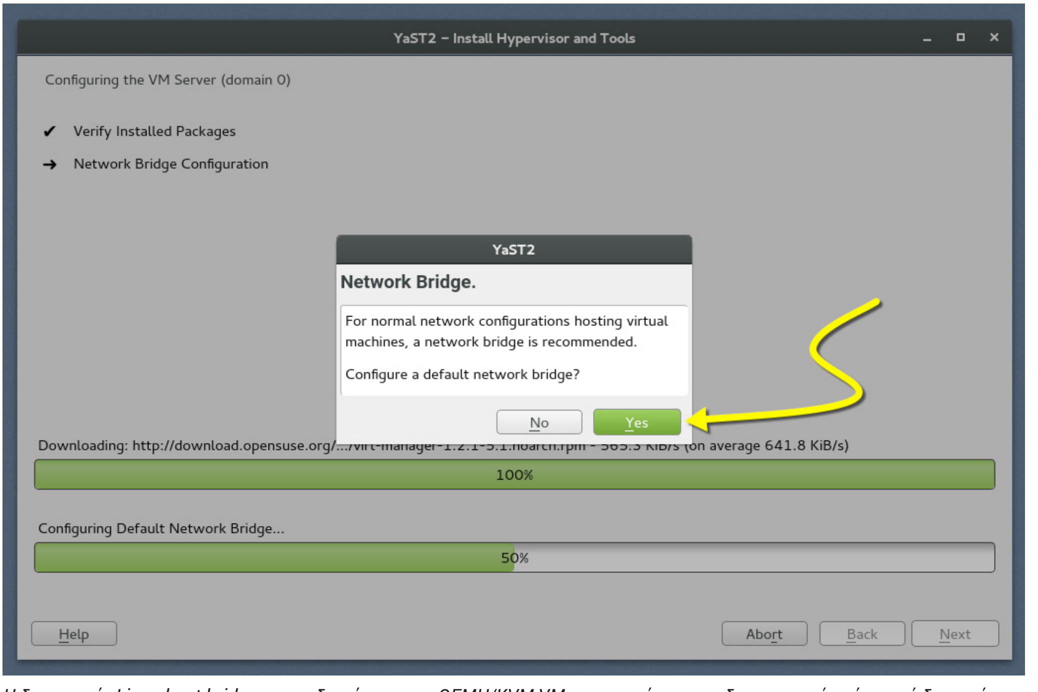
Σε πολύ λίγο θα καταφύγουμε στο εργαλείο virsh, προκειμένου να δούμε πώς μας βοηθά στη δημιουργία αλλά και στη διαχείριση δικτύων για εικονικές μηχανές.

Δύο μέθοδοι δικτύωσης

Ανεξαρτήτως του hypervizor, για το libvirt υπάρχουν δύο βασικές μέθοδοι δικτύωσης των εικονικών μηχανών: **Bridged Networking** και **NAT Forwarding**. Καθένα από τα δύο αυτά networking setups έχει τα υπέρ και τα κατά του.

Bridged Networking. Μέσω μίας ή περισσοτέρων **φυσικών** καρτών Ethernet, τα VMs (guests) με δικτύωση του είδους βλέπουν **απευθείας** το LAN εντός του οποίου βρίσκεται το host computer. (Ακριβέστερα, το βλέπουν μέσω του Linux host bridge. Περί bridging διαβάστε περισσότερα στο https://deltahacker.gr/?p=10500.) Το συγκεκριμένο setup προτείνεται για virtualization servers κι όχι, π.χ., για laptops, τα οποία

συνήθως μετακινούνται από δίκτυο σε δίκτυο. Οι μηχανές με bridged networking παρουσιάζουν τις καλύτερες δυνατές δικτυακές επιδόσεις, ενώ είναι κι εύκολα προσβάσιμες από άλλα (εικονικά ή φυσικά) μηχανήματα. Αν όμως το host computer βγαίνει στον έξω κόσμο μέσω ασύρματης σύνδεσης, το αντίστοιχο wireless interface δεν είναι δυνατόν να συμμετέχει σε ένα Linux host bridge. Σε περιπτώσεις σαν αυτή δεν μπορούμε να χρησιμοποιήσουμε Bridged Networking για τα guests. Ευτυχώς, υπάρχει και το NAT Forwarding.



Η δημιουργία Linux host bridge για τη δικτύωση των QEMU/KVM VMs επιτυγχάνεται με διαφορετικό τρόπο από διανομή σε διανομή. Αν πάντως το host OS του υπολογιστή μας είναι το openSUSE κι εγκαθιστούμε το λογισμικό για την υποστήριξη του KVM με τη βοήθεια του YaST2, μπορούμε να ζητήσουμε την αυτόματη δημιουργία bridge.

NAT Forwarding. Μετά την εγκατάσταση του libvirt και φυσικά την εκκίνηση της αντίστοιχης υπηρεσίας, στο host computer υπάρχει έτοιμο ένα **εικονικό network switch**. Τα VMs που συνδέονται πάνω του έχουν τις δικές τους, εσωτερικές διευθύνσεις IP. Χάρη στην τεχνική του NAT (Network Address Translation), όταν βγαίνουν στον έξω κόσμο εμφανίζονται με το δημόσιο IP του host computer. Προφανώς, ο συγκεκριμένος τρόπος δικτύωσης είναι ό,τι πρέπει για hosts που είτε αλλάζουν συνέχεια δίκτυο είτε συνδέονται στο Internet μέσω wireless interface. Αλλά αυτή η ευελιξία ή φορητότητα, αν θέλετε, έχει και το κόστος της: εικονικοί ή φυσικοί υπολογιστές που είναι **εξωτερικοί** ως προς τον virtualization host, δεν είναι σε θέση να ξεκινούν συνδέσεις με VMs πίσω από το εικονικό switch. Μόνος τρόπος για να το πετυχαίνουν είναι με την εισαγωγή κανόνων **port forwarding** στο firewall του host. Κάτι

τέτοιο, όμως, αυξάνει την πολυπλοκότητα του network setup για τον host και, όπως γίνεται φανερό, από ένα σημείο κι έπειτα δεν αξίζει να έχουμε server VMs πίσω από εικονικά network switches.

Το προκαθορισμένο εικονικό τοπικό δίκτυο

Αναφέραμε ότι με την εκκίνηση της υπηρεσίας του libvirt, στο host computer υπάρχει έτοιμο ένα εικονικό switch. Το host OS συνδέεται πάνω του μέσω του network bridge interface με όνομα virbr0. Στο πλαίσιο της παρουσίασής μας δουλεύουμε σε ένα laptop με openSUSE Leap 42.3 (hostname: dalvik), οπότε δείτε τώρα το virbr0 μέσα από το προγραμματάκι ip:

```
[cvar AT dalvík] ~> ip a show virbr0
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue [...]
link/ether 52:54:00:f6:ff:9c brd ff:ff:ff:ff:ff:ff
inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
valid_lft forever preferred_lft forever
```

Η διεύθυνση IP του virbr0 είναι 192.168.122.1, κι αυτή είναι η διεύθυνση του router για τις εικονικές μηχανές πίσω από το switch. Τώρα, όλα τα εικονικά τοπικά δίκτυα που ορίζουν τα network switches του libvirt –αρχικά ξεκινάμε με μόνο ένα τέτοιο switch–, τα βλέπουμε ανά πάσα στιγμή με τη βοήθεια του εργαλείου virsh:

```
[cvar AT dalvík] ~> virsh net-list
Name                 State      Autostart     Persistent
--------------------------------------------------------
default              active     yes           yes
```

Καμία έκπληξη εδώ: έχουμε ένα μόνο εικονικό δίκτυο με **όνομα** default, το οποίο είναι **ενεργό** (State = active), **μη-εφήμερο** (Persistent = yes, δηλαδή διατηρείται και μετά από επανεκκίνηση του libvirt), και ρυθμισμένο ώστε να **ενεργοποιείται αυτόματα** κατά την (επαν)εκκίνηση του libvirt (Autostart = yes). Συνοπτικές πληροφορίες για ένα συγκεκριμένο εικονικό τοπικό δίκτυο, όπως είναι το default που ήδη έχουμε, παίρνουμε κι έτσι:

```
[cvar AT dalvík] ~> virsh net-info default
Name:           default
UUID:           112ac4cc-79c0-4bc2-ae33-f7e068c15111
Active:         yes
Persistent:     yes
Autostart:      yes
```

```
Bridge:         virbr0
```

Κάθε εικονικό τοπικό δίκτυο στον κόσμο του libvirt ορίζεται από ένα αρχείο XML. Προβάλλοντας το XML του default, παίρνουμε αναλυτικότερες πληροφορίες για το αντίστοιχο δίκτυο. Δείτε πώς ακριβώς βλέπουμε το XML του default:

```
[cvar AT dalvík] ~> virsh net-dumpxml default
<network>
  <name>default</name>
  <uuid>112ac4cc-79c0-4bc2-ae33-f7e068c15111</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr0' stp='on' delay='0'/>
  <mac address='52:54:00:f6:ff:9c'/>
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254'/>
    </dhcp>
  </ip>
</network>
```

Παρατηρήστε ότι παρέχεται δυναμική απόδοση διευθύνσεων IP, ξεκινώντας από την 192.168.122.2 και φτάνοντας έως και την 192.168.122.254. Στην πραγματικότητα, για κάθε εικονικό τοπικό δίκτυο που θέλουμε υπηρεσίες DHCP και DNS, το libvirt ξεκινά ένα κατάλληλα ρυθμισμένο instance του dnsmasq. Πώς βλέπουμε τις αποδόσεις διευθύνσεων IP για όλα τα VMs που βρίσκονται εντός συγκεκριμένου εικονικού τοπικού δικτύου; Προς το παρόν έχουμε μόνον ένα τέτοιο δίκτυο, το default. Έχουμε επίσης ξεκινήσει ένα VM, το qleap, το οποίο είναι συνδεδεμένο πάνω στο virbr0 switch. Τη μία και μοναδική απόδοση διεύθυνσης από πλευράς dnsmasq, τη βλέπουμε με τη βοήθεια του virsh:

```
[cvar AT dalvík] ~> virsh net-dhcp-leases default
Expiry Time          MAC address       [...] IP address       Hostname [...]
-----------------------------------------------------------------------------
2017-10-29 10:42:57 52:54:00:45:67:31 [...] 192.168.122.86/24 qleap    [...]
```

Το VM του παραδείγματός μας, λοιπόν, έχει τη διεύθυνση 192.168.122.86. Με βάση αυτή μπορούμε να φτάνουμε στο qleap από το host computer, π.χ., ξεκινώντας μια σύνδεση SSH:

```
[cvar AT dalvík] ~> ssh userson@192.168.122.86
Password:
Last login: Sun Oct 29 08:16:22 2017 from 192.168.122.1
Have a lot of fun...
userson@qleap:~> hostname -f
qleap
```

Προς το παρόν δεν μπορούμε να φτάνουμε στο qleap με βάση το hostname ή το FQDN του (Fully Qualified Domain Name), αλλά λίγο αργότερα θα το τακτοποιήσουμε κι αυτό το θέμα.

Δημιουργία νέου εικονικού τοπικού δικτύου

Αναλόγως του πρότζεκτ πάνω στο οποίο δουλεύουμε, είναι πιθανό να μη θέλουμε το προκαθορισμένο δίκτυο default αλλά να επιθυμούμε τη δημιουργία κάποιου άλλου. Πάντα με τη βοήθεια του εργαλείου virsh, ας δημιουργήσουμε ένα νέο εικονικό τοπικό δίκτυο με όνομα delta, το οποίο εξυπηρετεί το domain delta.is και στα VMs παρέχει υπηρεσίες DHCP και DNS. Το νέο δίκτυο θα υποστηρίζει και NAT forwarding, επομένως τα VMs του θα βγαίνουν στον έξω κόσμο με το δημόσιο IP του host computer. Χρειαζόμαστε ένα αρχείο XML που θα περιγράφει τα χαρακτηριστικά του νέου αυτού δικτύου. Φτιάχνουμε ένα νέο αρχείο με όνομα delta.xml (στην πραγματικότητα δεν έχει σημασία το πώς θα τ' ονομάσουμε) και με το ακόλουθο περιεχόμενο:

```
<network>
  <name>delta</name>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <domain name='delta.is' localOnly="yes"/>
  <ip address='10.20.30.254' netmask='255.255.255.0'>
    <dhcp>
      <range start='10.20.30.200' end='10.20.30.253'/>
    </dhcp>
  </ip>
```

```
</network>
```

Παρατηρήστε ότι:

- το όνομα του νέου δικτύου είναι delta (οδηγία <name>delta</name>)
- υποστηρίζεται το NAT forwarding (ενότητα <forward mode='nat'> έως </forward>)
- το domain name για το dnsmasq είναι το delta.is (οδηγία <domain name='delta.is' localOnly="yes"/>)
- εξαιτίας της τιμής που αναθέσαμε στην ιδιότητα localOnly, τα ερωτήματα που αφορούν στο delta.is θα απαντώνται **μόνο** από τον DNS server του domain και **δεν** θα προωθούνται σε άλλους name servers, πιο ψηλά στην ιεραρχία του DNS (το domain delta.is το χρησιμοποιούμε αυθαίρετα, χωρίς καν να έχουμε ελέγξει αν υπάρχει στο Internet: αν πράγματι υπάρχει, δεν θέλουμε να απασχολούμε τον name server του με ερωτήματα που τις περισσότερες φορές αφορούν σε τοπικά VMs)
- για τα VMs του delta η διεύθυνση των router και name server είναι 10.20.30.254, το network mask είναι το 255.255.255.0, ενώ λαμβάνουν δυναμικά διευθύνσεις IP ξεκινώντας από την 10.20.30.200 και φτάνοντας έως και την 10.20.30.253 (ενότητα <ip address='10.20.30.254' netmask='255.255.255.0'> έως </ip>)

Προκειμένου να **ορίσουμε** το νέο μας δίκτυο, απλά πληκτρολογούμε:

```
[cvar AT dalvík] ~> virsh net-define delta.xml
Network delta defined from delta.xml
```

Ωραία, το μήνυμα που επέστρεψε το virsh είναι ενθαρρυντικό. Ας δούμε όλα τα εικονικά τοπικά δίκτυα στο σύστημά μας:

```
[cvar AT dalvík] ~> virsh net-list
Name                 State      Autostart      Persistent
----------------------------------------------------------
default              active     yes            yes
```

Βλέπουμε μόνο το default κι όχι το delta που μόλις ορίσαμε –κατά τα λεγόμενα του virsh– επιτυχώς. Το νέο δίκτυο είναι πράγματι ορισμένο, αλλά προς το παρόν είναι **ανενεργό**. Αν θέλουμε το virsh να επιστρέφει όλα τα δίκτυα, ενεργά κι ανενεργά, τότε στην εντολή net-list παρέχουμε και την παράμετρο --all:

```
[cvar AT dalvík] ~> virsh net-list --all
Name                 State      Autostart      Persistent
```

```
----------------------------------------------------------
default              active     yes           yes
delta                inactive   no            yes
```

Το δίκτυο delta το ενεργοποιούμε πληκτρολογώντας…

```
[cvar AT dalvík] ~> virsh net-start delta
Network delta started
```

…και φροντίζουμε ώστε να ξεκινά αυτόματα:

```
[cvar AT dalvík] ~> virsh net-autostart delta
Network delta marked as autostarted
```

Δείτε τώρα την έξοδο του virsh net-list, χωρίς την παράμετρο --all:

```
[cvar AT dalvík] ~> virsh net-list
Name                 State      Autostart     Persistent
----------------------------------------------------------
default              active     yes           yes
delta                active     yes           yes
```

Πάρα πολύ ωραία. Στην εντολή net-define του virsh δώσαμε το αρχείο delta.xml και με βάση αυτό το εργαλείο παρήγαγε ένα εκτενέστερο αρχείο XML. Δείτε τι εννοούμε:

```
[cvar AT dalvík] ~> virsh net-dumpxml delta
<network>
  <name>delta</name>
  <uuid>e1a9536a-d185-4c04-ab63-459762232839</uuid>
  <forward mode='nat'>
    <nat>
      <port start='1024' end='65535'/>
    </nat>
  </forward>
  <bridge name='virbr1' stp='on' delay='0'/>
  <mac address='52:54:00:12:e1:d7'/>
  <domain name='delta.is' localOnly="yes"/>
```

```
  <ip address='10.20.30.254' netmask='255.255.255.0'>
    <dhcp>
      <range start='10.20.30.200' end='10.20.30.253'/>
    </dhcp>
  </ip>
</network>
```

Αμέσως διαπιστώνουμε ότι το virsh είναι αρκετά έξυπνο, ώστε να δημιουργεί αυτόματα τα ακόλουθα στοιχεία:

- ένα μοναδικό UUID για το νέο δίκτυο (<uuid>e1a9536a-d185-4c04-ab63-459762232839</uuid>)
- ένα νέο network bridge interface, το virbr1, για το host OS
- ένα μοναδικό MAC address για το νέο bridge (<mac address='52:54:00:12:e1:d7'/>)

Ιδού και τα network bridge interfaces των δύο εικονικών τοπικών δικτύων, όπως φαίνονται από το host OS:

```
[cvar AT dalvík] ~> ip a show
[...]
10: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue [...]
link/ether 52:54:00:f6:ff:9c brd ff:ff:ff:ff:ff:ff
inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
valid_lft forever preferred_lft forever
[...]
17: virbr1: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue [...]
link/ether 52:54:00:12:e1:d7 brd ff:ff:ff:ff:ff:ff
inet 10.20.30.254/24 brd 10.20.30.255 scope global virbr1
valid_lft forever preferred_lft forever
[...]
```

Επιλογή δικτύου κατά τη δημιουργία νέου QEMU/KVM VM

Σε σχετικά άρθρα του τεύχους 060, τα οποία μπορείτε να διαβάσετε στο https://deltahacker.gr/?p=17646 και στο https://deltahacker.gr/?p=17684, δείξαμε πώς δημιουργούμε QEMU/KVM VMs από τη γραμμή εντολών. Ουσιαστικά καταφεύγουμε στο εργαλείο virt-install και το δίκτυο στο οποίο θα ανήκει το νέο VM το δίνουμε ως όρισμα στην παράμετρο --network. Για λόγους πληρότητας, ας δούμε άλλο ένα παράδειγμα δημιουργίας VM με guest OS το Fedora 26 Server, το οποίο θα βρίσκεται εντός του δικτύου delta. Αρχικά φτιάχνουμε ένα νέο QCOW2 image μεγέθους 128GiB, το οποίο θα αποτελέσει το δίσκο εγκατάστασης του VM:

```
[cvar AT dalvík] ~> virsh vol-create-as theQCOW2s \
> --name fedora26.qcow2 --capacity 128G --format qcow2
```

Το νέο image είναι sparse file (το μέγεθός του δεν είναι 128GiB, απλά αυτό είναι το άνω όριο για τη δυναμική επέκταση του αρχείου), βρίσκεται δε μέσα στο pool με όνομα theQCOW2s. Αμέσως μετά προχωράμε στη δημιουργία του VM:

```
[cvar AT dalvík] ~> virt-install --name fedora26 \
> --memory 1536 --cpu host --vcpus=2 \
> --network network=delta,model=virtio \
> --disk ~/theQCOW2s/fedora26.qcow2,bus=virtio \
> --location over_there
```

Το νέο μας VM ονομάζεται fedora26, έχει 1.5GiB μνήμης RAM, δύο εικονικούς επεξεργαστές και συνδέεται στο switch του δικτύου delta. Ενδιαφέρον έχει η παράμετρος --location, με χρήση της οποίας λέμε ότι η εγκατάσταση θα γίνει δικτυακά και συγκεκριμένα από το mirror στο Πανεπιστήμιο της Κρήτης, στη διεύθυνση https://ftp.cc.uoc.gr/pub/linux/fedora/linux/releases/26/Server/x86_64/os (αντικαταστήστε το over_there με τη διεύθυνση που μόλις δώσαμε). Πληκτρολογώντας την παραπάνω εντολή το VM δημιουργείται, ενεργοποιείται και, αφού κατέβουν ορισμένα απαραίτητα αρχεία, ξεκινά η διαδικασία εγκατάστασης του Fedora 26 Server. Μετά την ολοκλήρωσή της, αν όλα έχουν πάει καλά η εντολή virsh net-dhcp-leases delta θα επιστρέψει το DHCP lease του VM:

```
[cvar AT dalvík] ~> virsh net-dhcp-leases delta
Expiry Time         MAC address       [...] IP address      Hostname [...]
--------------------------------------------------------------------------
2017-10-29 17:32:16 52:54:00:ad:7e:49 [...] 10.20.30.200/24 fedora26 [...]
```

Ας συνδεθούμε για πρώτη φορά από το host computer μέσω SSH, στο λογαριασμό του root:

```
[cvar AT dalvík] ~> ssh root@10.20.30.200
The authenticity of host '10.20.30.200 (10.20.30.200)' can't be established.
[...]
Are you sure you want to continue connecting (yes/no)? yes
[...]
root@10.20.30.200's password:
[root@fedora26 ~]# hostname -f
fedora26.delta.is
```

Όπως βλέπετε, το hostname που επιλέξαμε για το VM μας είναι το (καθόλου πρωτότυπο) fedora26. Το δε domain, το οποίο κατά τα αναμενόμενα είναι το delta.is, δεν το δώσαμε εμείς αλλά ελήφθη από το instance του dnsmasq για το εικονικό τοπικό δίκτυο delta.

Αλλαγή δικτύου για υπάρχον QEMU/KVM VM

Είναι πολύ εύκολο να αλλάξουμε το εικονικό τοπικό δίκτυο στο οποίο ανήκει ένα συγκεκριμένο VM. Στο πλαίσιο της παρουσίασής μας, για παράδειγμα, έχουμε το VM με όνομα qleap, από το δίκτυο default. Προκειμένου να το μεταφέρουμε στο δίκτυο delta, φροντίζουμε ώστε το VM να είναι ανενεργό (αν **δεν** είναι δίνουμε virsh shutdown qleap) και τροποποιούμε το XML του πληκτρολογώντας virsh edit qleap. Το περιεχόμενο του XML θα φορτώσει στον προκαθορισμένο editor και θα εντοπίσουμε τη γραμμή <source network='default'/>. Το μόνο που χρειάζεται ν' αλλάξουμε είναι το default, το οποίο θα το κάνουμε delta. Αποθηκεύουμε την αλλαγή, εγκαταλείπουμε τον editor κι αν δεν έχουμε εισαγάγει κάποιο συντακτικό λάθος στο XML, το virsh λέει: Domain qleap XML configuration edited. Αν τώρα ξεκινήσουμε το VM (virsh start qleap), μετά από λίγο θα το δούμε κι αυτό στη λίστα με τα DHCP leases για τα VMs εντός του delta:

```
[cvar AT dalvík] ~> virsh net-dhcp-leases delta
Expiry Time         MAC address       Protocol IP address      Hostname [...]
-------------------------------------------------------------------------------
2017-10-29 20:16:16 52:54:00:45:67:31 [...]    10.20.30.231/24 qleap    [...]
2017-10-29 20:00:37 52:54:00:ad:7e:49 [...]    10.20.30.200/24 fedora26 [...]
```

Τροποποίηση υπάρχοντος εικονικού τοπικού δικτύου

Τα αρχεία XML των εικονικών τοπικών δικτύων βρίσκονται κάτω από τον κατάλογο /etc/libvirt/qemu/networks:

```
[cvar AT dalvík] ~> sudo ls -lh /etc/libvirt/qemu/networks
total 8.0K
drwx------ 1 root root  40 Oct 29 12:23 autostart
-rw------- 1 root root 576 Apr  8  2017 default.xml
-rw------- 1 root root 667 Oct 29 12:05 delta.xml
```

Επιπρόσθετα, για τα δίκτυα που ξεκινούν αυτόματα υπάρχει κι από ένα symbolic link μέσα στον κατάλογο /etc/libvirt/qemu/networks/autostart. Για το σύστημά μας έχουμε δύο δίκτυα, τα default και delta, τα οποία αμφότερα ξεκινούν αυτόματα. Μέσα στο autostart, λοιπόν, θα πρέπει να υπάρχουν δύο links. Πράγματι, έτσι είναι:

```
[cvar AT dalvík] ~> sudo ls -lh /etc/libvirt/qemu/networks/autostart
total 8.0K
[...] Sep 12 20:32 default.xml -> /etc/libvirt/qemu/networks/default.xml
[...] Oct 29 12:23 delta.xml -> /etc/libvirt/qemu/networks/delta.xml
```

Τώρα, αν θέλουμε να τροποποιήσουμε τις ιδιότητες ενός συγκεκριμένου δικτύου, τότε **δεν** επεμβαίνουμε απευθείας στο αντίστοιχο αρχείο XML. Αντίθετα, το τροποποιούμε δίνοντας στο virsh την εντολή net-edit ακολουθούμενη από το **όνομα** του δικτύου κι **όχι** από το πλήρες όνομα του αντίστοιχου αρχείου XML. Ως ένα παράδειγμα, ας δούμε πώς λέμε στο στο dnsmasq instance του default να εξυπηρετεί το domain με όνομα local.kvm. Από το host OS ανοίγουμε το XML του δικτύου default:

```
[cvar AT dalvík] ~> virsh net-edit default
```

Ανάμεσα σ' αυτές τις δύο γραμμές...

```
<mac address='52:54:00:f6:ff:9c'/>
<ip address='192.168.122.1' netmask='255.255.255.0'>
(το MAC address θα διαφέρει στον δικό σας υπολογιστή) βάζουμε αυτή τη γραμμή:
<domain name='local.kvm' localOnly="yes"/>
```

Πρέπει να έχουμε, δηλαδή, αυτές τις τρεις γραμμές:

```
<mac address='52:54:00:f6:ff:9c'/>
<domain name='local.kvm' localOnly="yes"/>
<ip address='192.168.122.1' netmask='255.255.255.0'>
```

Αποθηκεύουμε τις αλλαγές στο XML, εγκαταλείπουμε τον editor και το virsh λέει: Network default XML configuration edited. Αν όμως σκεφτούμε να ελέγξουμε για την αλλαγή, γράφοντας virsh net-dumpxml default, δεν θα τη δούμε:

```
[cvar AT dalvík] ~> virsh net-dumpxml default
[...]
<mac address='52:54:00:f6:ff:9c'/>
<ip address='192.168.122.1' netmask='255.255.255.0'>
[...]
```

Δεν έχει ληφθεί ακόμη υπόψη και χρειάζεται να επανεκκινήσουμε το δίκτυο default: το σταματάμε πληκτρολογώντας virsh net-destroy default (ναι, σ' αυτή την περίπτωση το "destroy" σημαίνει "stop"), κι αμέσως μετά το ξεκινάμε και πάλι με ένα virsh net-start default. Δείτε τώρα τι ισχύει:

```
[cvar AT dalvík] ~> virsh net-dumpxml default
[...]
<mac address='52:54:00:f6:ff:9c'/>
<domain name='local.kvm' localOnly="yes"/>
<ip address='192.168.122.1' netmask='255.255.255.0'>
[...]
```

Ας κάνουμε κι έναν έλεγχο μ' ένα VM "κάτω" από το local.kvm. Πριν από λίγο μεταφέραμε τη μηχανή qleap από το δίκτυο default στο δίκτυο delta. Εργαζόμενοι με παρόμοιο τρόπο τη μεταφέρουμε πίσω στο default -- και φυσικά την ενεργοποιούμε γράφοντας virsh start qleap. Σε ελάχιστα δευτερόλεπτα θα υπάρχει ένα νέο DHCP lease για το qleap:

```
[cvar AT dalvík] ~> virsh net-dhcp-leases default
Expiry Time         MAC address       [...] IP address        Hostname [...]
-----------------------------------------------------------------------------
2017-10-29 23:42:30 52:54:00:45:67:31 [...] 192.168.122.85/24 qleap    [...]
```

Συνδεόμαστε στη μηχανή (ssh userson@192.168.122.85) κι ελέγχουμε το FQDN του συστήματος:

```
userson@qleap:~> hostname -f
qleap.local.kvm
```

Όλα καλά, το FQDN του qleap είναι το αναμενόμενο.

Resolving από το host OS

Θα παρατηρήσατε ότι έως τώρα συνδεόμασταν από τον host στα VMs με βάση τα IP. Πράγματι, είναι απαραίτητο να γνωρίζουμε τις αριθμητικές διευθύνσεις: τα hostnames ή τα FQDNs που έχουν οι μηχανές ενός εικονικού δικτύου **δεν** είναι ορατά στο host computer, αφού αυτό δεν έχει πρόσβαση στους name servers των αντίστοιχων τοπικών δικτύων.

Προκειμένου να μην πληκτρολογούμε IPs θα μπορούσαμε να βάζαμε στο /etc/hosts του host κατάλληλα entries, ώστε αντί για IPs να θυμόμαστε hostnames ή FQDNs. Μάλιστα τις πληροφορίες του /etc/hosts λαμβάνουν υπόψη και τα dnsmasq

instances για όλα τα εικονικά τοπικά δίκτυα, οπότε μια μηχανή, π.χ., από το default, είναι δυνατόν να φτάνει σε μια μηχανή του delta με βάση το hostname ή το FQDN της. Αλλά η χειροκίνητη ενημέρωση του /etc/hosts δεν είναι **καθόλου** βολική.

Μία σαφώς καλύτερη προσέγγιση είναι να φροντίσουμε ώστε οι επιθυμητοί name servers να υπάρχουν πάντα στο αρχείο /etc/resolv.conf του host. Σε ένα σύγχρονο laptop –ή ακόμη και σε ένα desktop computer– με Linux, τα θέματα της δικτύωσης τα διαχειρίζεται ο Network Manager. Η σχετική υπηρεσία μεταξύ άλλων δημιουργεί **αυτόματα** το /etc/resolv.conf. Έχουμε όμως τη δυνατότητα να υποδεικνύουμε τα domains ή/και τους name servers που **θέλουμε** να συμπεριλαμβάνονται στο resolv.conf, ανεξαρτήτως του δικτύου που τη δεδομένη στιγμή βρίσκεται ο υπολογιστής μας.

Στο laptop των δοκιμών μας, το οποίο όπως αναφέραμε τρέχει openSUSE Leap 42.3, υπάρχει φυσικά ο Network Manager. Θέλαμε να έχουμε πάντοτε τον name server του δικτύου delta στο /etc/resolv.conf του συστήματος, οπότε κάναμε μια μικρή επέμβαση στο αρχείο /etc/dhclient.conf. Συγκεκριμένα, φροντίσαμε ώστε να περιλαμβάνει τις ακόλουθες δύο γραμμές:

```
supersede domain-name "delta.is colder.xyz";
prepend domain-name-servers 10.20.30.254;
```

Η πρώτη σημαίνει ότι το resolv.conf θα έχει **πάντα** πάνω-πάνω την ακόλουθη οδηγία:

```
search delta.is colder.xyz
```

Έτσι, όταν πληκτρολογούμε μόνο το hostname ενός μηχανήματος, π.χ., randomhost, τότε **πρώτα** θα αναζητείται το IP του randomhost.delta.is και, αν δεν υπάρχει, μετά θα αναζητείται το IP του randomhost.colder.xyz. Το delta.is είναι το domain ενός από τα εικονικά τοπικά μας δίκτυα, ενώ το colder.xyz είναι ένα άλλο (δημόσιο) domain που έχουμε στη διάθεσή μας. Η δεύτερη γραμμή στο dhclient.conf σημαίνει ότι **ανεξαρτήτως** των name servers που θα πάρει ο Network Manager από τον DHCP server του εκάστοτε δικτύου, εντός του οποίου βρίσκεται το laptop, ο πρώτος name server που θα συμβουλεύεται το σύστημα θα είναι **πάντοτε** ο 10.20.30.254: αυτό ακριβώς είναι το IP που έχει ο router και name server του εικονικού τοπικού δικτύου με όνομα delta. Προκειμένου ο Network Manager να λάβει **αμέσως** υπόψη τις νέες οδηγίες στο /etc/dhclient.conf, κι επομένως να δημιουργήσει μια ανανεωμένη εκδοχή του /etc/resolv.conf, αρκεί να το επανεκκινήσουμε:

```
[cvar AT dalvík] ~> sudo systemctl restart NetworkManager
```

Ιδού πώς μοιάζει το /etc/resolv.conf του laptop μας, όταν είναι συνδεδεμένο σε ένα ασύρματο δίκτυο κάπου στο Λευκαντί Ευβοίας:

```
[cvar AT dalvík] ~> cat /etc/resolv.conf
# Generated by NetworkManager
search delta.is colder.xyz
nameserver 10.20.30.254
nameserver 192.168.1.1
```

Ο πρώτος name server είναι εκείνος του δικτύου delta. Ο δεύτερος είναι εκείνος του modem/router με WiFi access point, σε μια καφετέρια κάπου στο Λευκαντί. Δείτε και το resolv.conf όταν το laptop είναι συνδεδεμένο στο οικιακό μας δίκτυο:

```
[cvar AT dalvík] ~> cat /etc/resolv.conf
# Generated by NetworkManager
search delta.is colder.xyz
nameserver 10.20.30.254
nameserver 192.168.178.2
```

Το μόνο που αλλάζει τώρα είναι η διεύθυνση IP του δεύτερου name server – και οι προσεκτικοί αναγνώστες αμέσως θα καταλάβουν ότι πρόκειται για τη διεύθυνση του Raspberry Pi μας, το οποίο παρέχει υπηρεσίες network-wide ad-blocker (https://deltahacker.gr/?p=17672). Όπου κι αν βρισκόμαστε με το laptop μας, χάρη στις αλλαγές που κάναμε στο /etc/dhclient.conf μπορούμε να αναφερόμαστε σε VMs του delta είτε με βάση το hostname τους είτε με βάση το FQDN τους. Παράδειγμα:

```
[cvar AT dalvík] ~> ping -c 3 fedora26
PING fedora26.delta.is (10.20.30.200) 56(84) bytes of data.
64 bytes from fedora26.delta.is (10.20.30.200): icmp_seq=1 ttl=64 [...]
64 bytes from fedora26.delta.is (10.20.30.200): icmp_seq=2 ttl=64 [...]
64 bytes from fedora26.delta.is (10.20.30.200): icmp_seq=3 ttl=64 [...]
--- fedora26.delta.is ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.141/0.155/0.164/0.010 ms
```

Μια χαρά.

Τροποποίηση συμπεριφοράς dnsmasq instance

Είναι πιθανό να επιθυμούμε την αλλαγή της συμπεριφοράς του dnsmasq instance για ένα συγκεκριμένο εικονικό τοπικό δίκτυο. Ας υποθέσουμε, για παράδειγμα, ότι το fedora26.delta.is θέλουμε να παίρνει πάντοτε το ίδιο IP, συγκεκριμένα το 10.20.30.5. Χρειαζόμαστε το MAC address της κάρτας Ethernet του VM, κι ένας τρόπος για να το

μάθουμε είναι παρατηρώντας την έξοδο του virsh net-dhcp-leases delta (στο πλαίσιο της παρουσίασης μας, το ζητούμενο MAC address είναι το 52:54:00:ad:7e:49). Στη συνέχεια πληκτρολογούμε:

```
[cvar AT dalvík] ~> virsh net-update delta add ip-dhcp-host \
> "<host mac='52:54:00:ad:7e:49' ip='10.20.30.5' />" --live --config
```

Το virsh επιστρέφει το μήνυμα Updated network delta persistent config and live state, που σημαίνει ότι η αλλαγή ελήφθη υπόψη και δεν χρειάζεται επανεκκίνηση του δικτύου delta. Επανεκκινούμε μόνο τη μηχανή (virsh reboot fedora26), συνδεόμαστε μέσω SSH (ssh root@fedora26.delta.is) κι αμέσως ελέγχουμε τη διεύθυνση IP που έχει το ένα και μοναδικό Ethernet interface, το ens3:

```
[root@fedora26 ~]# ip a show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel [...]
link/ether 52:54:00:ad:7e:49 brd ff:ff:ff:ff:ff:ff
inet 10.20.30.5/24 brd 10.20.30.255 scope global dynamic ens3
valid_lft 3118sec preferred_lft 3118sec
inet6 fe80::5054:ff:fead:7e49/64 scope link
valid_lft forever preferred_lft forever
```

Τέλεια όλα.

Κανόνες port forwarding στο host computer

Το παράδειγμα που μόλις δώσαμε, με το static IP mapping, δεν ήταν εντελώς τυχαίο. Κάποιες φορές θα θέλουμε ένα VM, το οποίο βρίσκεται εντός ενός εικονικού τοπικού δικτύου, να είναι προσβάσιμο κι από μηχανήματα που είναι εξωτερικά ως προς το host computer (εν προκειμένω, το laptop με το openSUSE). Σε τέτοιες περιπτώσεις βοηθά το VM να έχει στατική IP. Φροντίσαμε πριν λίγο ώστε η εικονική μηχανή fedora26, του δικτύου delta, να παίρνει πάντοτε το ίδιο IP (το 10.20.30.5). Μετά την εγκατάσταση του Fedora 26 Server, εξάλλου, είναι ενεργοποιημένη μία υπηρεσία για τη διαχείριση (ή καλύτερα για την εποπτεία) του συστήματος μέσω web. Πράγματι, αν από έναν browser του host computer επισκεφτούμε τη διεύθυνση https://10.20.30.5:9090, θα μας ζητηθεί να συνδεθούμε σε έναν υπάρχοντα λογαριασμό χρήστη του συστήματος. Ως ένα απλό παράδειγμα εισαγωγής κανόνων port forwarding για ένα QEMU/KVM VM **πίσω** από ένα εικονικό τοπικό δίκτυο, ας δούμε πώς είναι δυνατόν να φτάσουμε στην ίδια σελίδα από ένα άλλο laptop, με macOS. Για το τοπικό δίκτυο που ορίζει το WiFi access point που τώρα είμαστε συνδεδεμένοι, το πλήρες όνομα του laptop με το openSUSE είναι dalvik.lan (το πλήρες όνομα του laptop με το macOS είναι mbpr15.lan, όμως αυτό δεν έχει καμία σημασία). Από έναν web browser του laptop με το macOS, όποτε πηγαίνουμε στο https://dalvik.lan:59090 θα πρέπει να ανακατευθυνόμαστε στο https://10.20.30.5:9090. Στην περίπτωσή μας,

το port του host computer (dalvik.lan), που επιλέξαμε να είναι το 59090, θα μπορούσε να ταυτίζεται με το port του VM, που είναι το 9090. Σκόπιμα, όμως, αποφασίσαμε να τα διαφοροποιήσουμε.

Πριν κάνουμε κάτι, βεβαιωνόμαστε ότι το fedora26 είναι **απενεργοποιημένο**. Μετά, με δικαιώματα διαχειριστή φτιάχνουμε στον host το αρχείο qemu, κάτω από τον κατάλογο /etc/libvirt/hooks:

```
[cvar AT dalvík] ~> sudo touch /etc/libvirt/hooks/qemu
```

Ξανά με δικαιώματα διαχειριστή ανοίγουμε το qemu με τον text editor της προτίμησής μας. Γράφουμε εντός τον κώδικα BASH που ακολουθεί:

```
#!/bin/bash
IPT=/usr/sbin/iptables
if [ "${1}" = "fedora26" ]; then
  VM_IP=10.20.30.5
  VM_PORT=9090
  HOST_PORT=59090
  VIRT_NET=virbr1
  if [ "${2}" = "stopped" ] || [ "${2}" = "reconnect" ]; then
    $IPT -D FORWARD -o $VIRT_NET -d $VM_IP -j ACCEPT
    $IPT -t nat -D PREROUTING -p tcp --dport $HOST_PORT -j DNAT --to $VM_
IP:$VM_PORT
  fi
  if [ "${2}" = "start" ] || [ "${2}" = "reconnect" ]; then
    $IPT -I FORWARD -o $VIRT_NET -d $VM_IP -j ACCEPT
    $IPT -t nat -I PREROUTING -p tcp --dport $HOST_PORT -j DNAT --to $VM_
IP:$VM_PORT
  fi
fi
```

(Βασίζεται στο script από το επίσημο documentation του libvirt, https://wiki.libvirt.org/page/Networking#Forwarding_Incoming_Connections.) Για το δικό σας setup αλλάξτε τις μεταβλητές VM_IP, VM_PORT, HOST_PORT και VIRT_NET καταλλήλως, ενώ προσέξτε και τη μεταβλητή IPT. Αποθηκεύουμε, εγκαταλείπουμε τον editor και κάνουμε το αρχείο qemu εκτελέσιμο:

```
[cvar AT dalvík] ~> sudo chmod +x /etc/libvirt/hooks/qemu
```

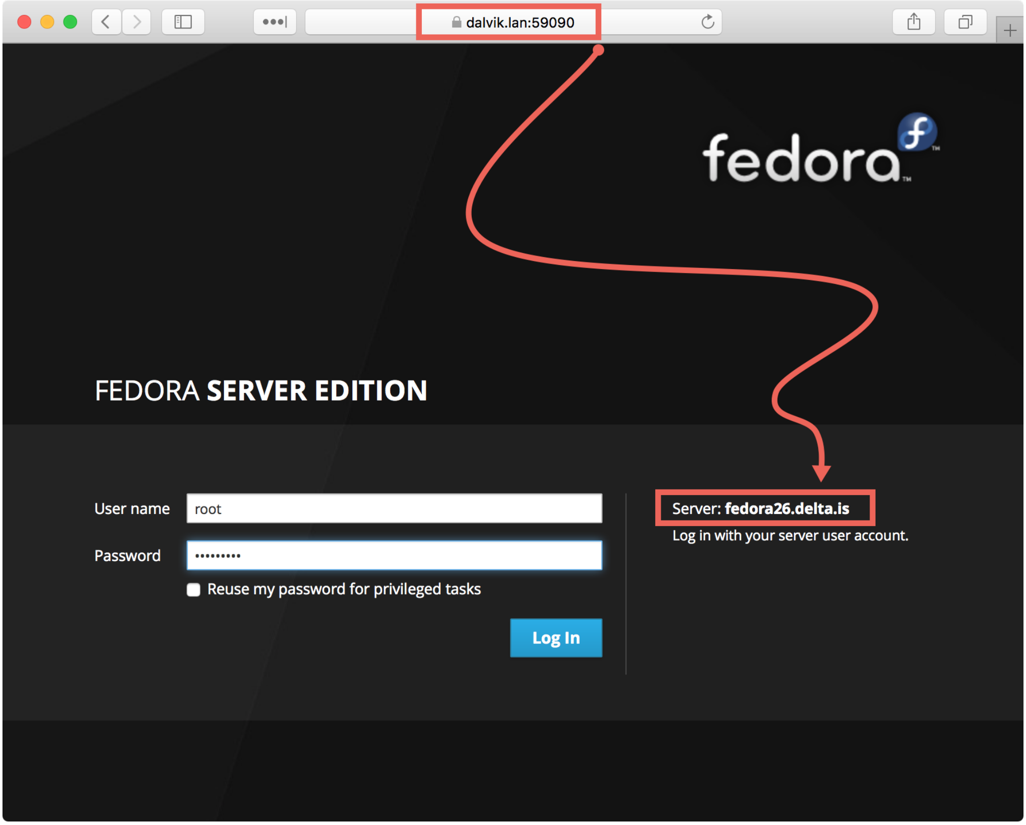
Επανεκκινούμε το libvirt…

```
[cvar AT dalvík] ~> sudo systemctl restart libvirtd
```

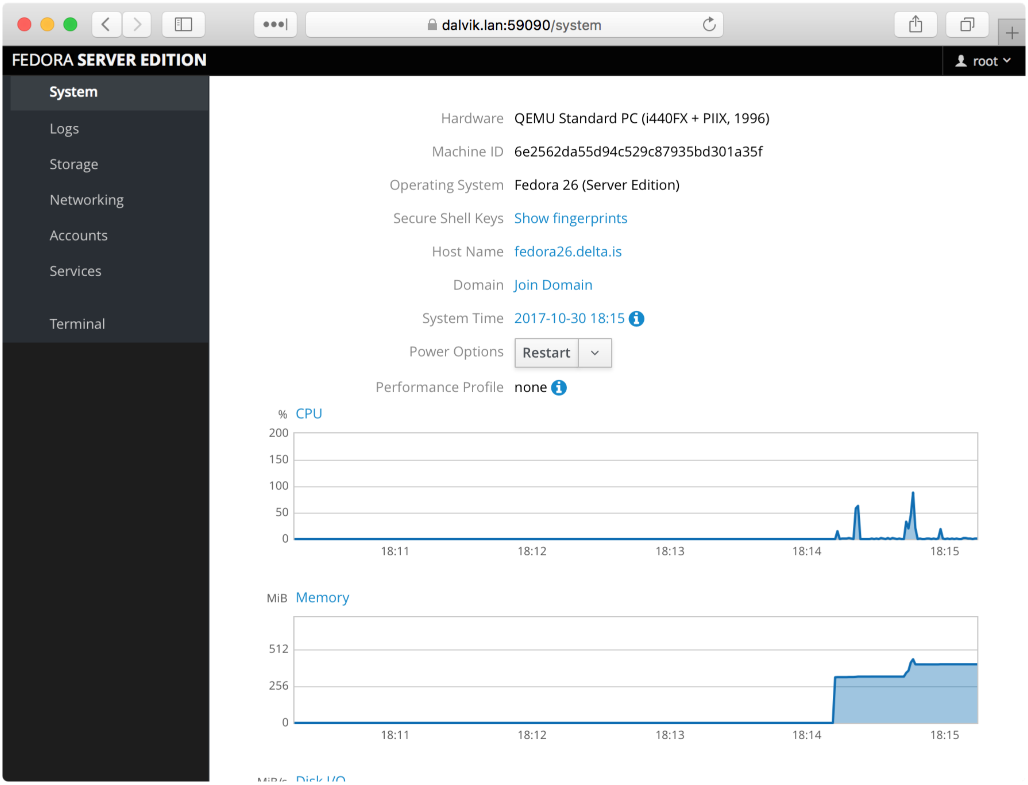
…και βεβαίως ξεκινάμε το VM:

```
[cvar AT dalvík] ~> virsh start fedora26
```

Αυτό ήταν. Από το άλλο laptop (αυτό με το macOS) ανοίγουμε έναν web browser και κατευθυνόμαστε στο https://dalvik.lan:59090. Αν όλα έχουν πάει καλά, θα λάβουμε προειδοποίηση για self-signed πιστοποιητικό. Το αποδεχόμαστε και προχωράμε. Δείτε τα δύο screenshots που ακολουθούν, διαβάστε και τις αντίστοιχες περιγραφές. Επίσης: Καλή σας διασκέδαση!



Από τον web browser ενός laptop με macOS συνδεόμαστε στο laptop με το openSUSE, το οποίο έχει ρόλο virtualization host. Χάρη στους κανόνες port forwarding που ισχύουν για το libvirt, ανακατευθυνόμαστε στο web panel ενός VM που βρίσκεται εντός ενός εικονικού τοπικού δικτύου και γενικά δεν είναι προσβάσιμο από τον έξω κόσμο.



Πρόσβαση στο web panel ενός εικονικού υπολογιστή με Fedora 26 Server, ο οποίος βρίσκεται εντός εικονικού τοπικού δικτύου και σε γενικές γραμμές δεν είναι προσβάσιμος από μηχανήματα ξένα ως προς τον virtualization host.


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

Skill: Beginner
Tags: DNS, security, privacy, anycast

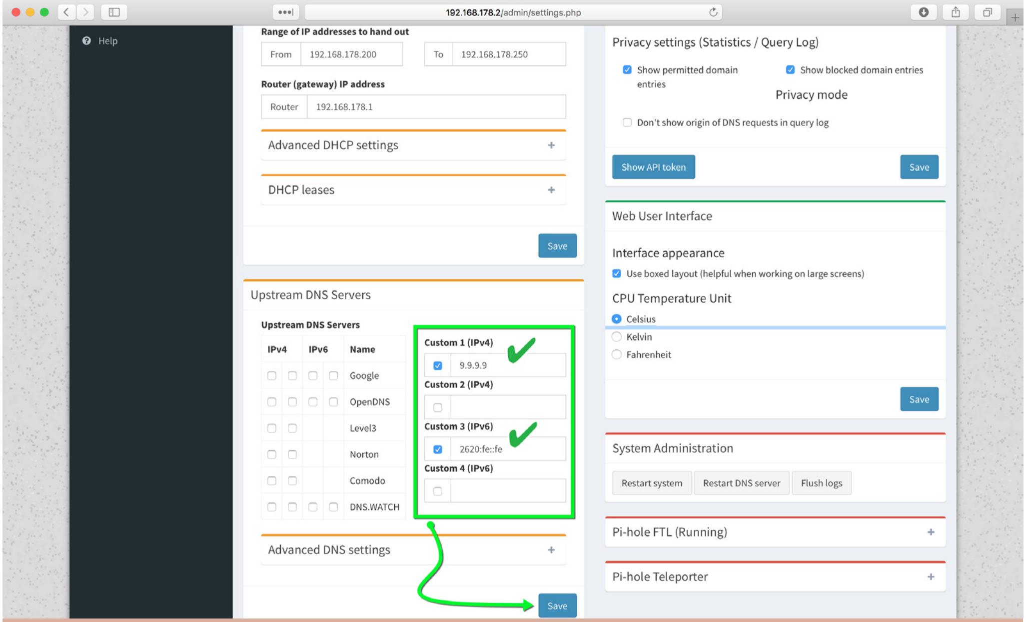
Υπηρεσία DNS με έμφαση στην ασφάλεια

Το Quad9 είναι μια νέα πλατφόρμα DNS από τις IBM, Packet Clearing House και Global Cyber Alliance, σχεδιασμένη με έμφαση την ιδιωτικότητα και κυρίως την ασφάλεια. Ανεξαρτήτως συσκευής ή λειτουργικού συστήματος, οποιοσδήποτε είναι σε θέση να χρησιμοποιεί το Quad9 και μάλιστα εντελώς δωρεάν.

Η βασική ιδέα πίσω από το Quad9 (https://quad9.net) είναι ότι οι χρήστες προστατεύονται αυτόματα από κακόβουλους δικτυακούς τόπους ή γενικότερα από κακόβουλους δικτυακούς **πόρους**. Κάθε φορά που κάνουμε κλικ σε link ή πληκτρολογούμε τη διεύθυνση προς κάποια τοποθεσία στο web, πριν το Quad9 επιστρέψει το αντίστοιχο IP ελέγχει αν ο ζητούμενος πόρος υπάρχει σε μια εκτενή βάση δεδομένων με κακόβουλες τοποθεσίες. Αν υπάρχει, τότε η πρόσβαση στον πόρο απλά μπλοκάρεται. Επιπλέον της προαναφερθείσας βάσης, το Quad9 κάνει χρήση των πληροφοριών που συλλέγουν συνεργάτες της IBM από το χώρο της ασφάλειας. Το Quad9 εξάλλου είναι υπέρ της ιδιωτικότητας (privacy), αφού οι διευθύνσεις IP των χρηστών δεν αποθηκεύονται σε κανένα από τα μηχανήματα που συγκροτούν το όλο σύστημα. Αυτά, παρεμπιπτόντως, είναι κατανεμημένα σε όλον τον πλανήτη. Ήδη υπάρχουν 100 και πλέον τοποθεσίες με servers του Quad9 κι εντός του 2018 ο σχεδιασμός προβλέπει περισσότερες από 160. Οι name servers δεν καλύπτουν μόνο μεγάλα αστικά κέντρα. Έτσι, όσοι βρίσκονται μακριά απ' αυτά πιθανώς να παίρνουν ταχύτερα απαντήσεις για τα DNS queries, σε σχέση με άλλες υπηρεσίες DNS.

Η διεύθυνση IPv4 του Quad9 DNS είναι η 9.9.9.9, ενώ η διεύθυνση IPv6 είναι η 2620:fe::fe. Χάρη στην τεχνολογία του **anycast**, αρκεί να στέλνουμε τα DNS queries πάντα προς την ίδια διεύθυνση IP κι αυτά θα ανακατευθύνονται αυτόματα προς τον γεωγραφικά πλησιέστερο server. Τώρα, υπάρχουν διάφοροι τρόποι ώστε ν' αρχίσουμε να χρησιμοποιούμε το Quad9 — ή έστω να το δοκιμάσουμε κι αξιολογήσουμε. Ο κλασικός είναι με χειροκίνητη ρύθμιση του υποσυστήματος δικτύωσης του λειτουργικού ενός υπολογιστή ή μιας συσκευής. Εναλλακτικά, ίσως θελήσουμε να επέμβουμε στις ρυθμίσεις του router στο τοπικό μας δίκτυο, ώστε ο τελευταίος να απευθύνει τα DNS queries των πελατών του στη διεύθυνση 9.9.9.9 (ή και στη διεύθυνση 2620:fe::fe).

Να αναφέρουμε τέλος και την περίπτωση του Pi-hole, το οποίο αρκετοί αναγνώστες του περιοδικού ίσως έχουν ήδη εντάξει στο οικιακό τους δίκτυο για το μπλοκάρισμα των διαφημίσεων (https://deltahacker.gr/?p=17672). Δείτε το ακόλουθο screenshot, διαβάστε και τη συνοδευτική περιγραφή.



Είναι πολύ εύκολο να ρυθμίσουμε το Pi-hole, ώστε να χρησιμοποιεί το Quad9 DNS system. Ξεκινάμε με login στο administration web panel του Pi-hole κι ακολουθούμε τη διαδρομή Settings -> Upstream DNS Servers. Εκεί αποεπιλέγουμε τους DNS servers που ήταν επιστρατευμένοι έως τώρα, κάνουμε κλικ στο κουμπάκι της θυρίδας Custom 1 (IPv4) και πληκτρολογούμε τη διεύθυνση 9.9.9.9. Προαιρετικά, κάνουμε κι ένα κλικ στο κουμπάκι της θυρίδας Custom 3 (IPv6) και δίνουμε τη διεύθυνση 2620:fe::fe. Επικυρώνουμε τις επιλογές πατώντας στο κουμπί Save, στο κάτω μέρος της σελίδας. Στο εξής, το Pi-hole θα χρησιμοποιεί το σύστημα DNS του Quad9. Αν το Pi-hole έχει και το ρόλο του DHCP server στο τοπικό δίκτυο, τότε το ίδιο σύστημα DNS θα χρησιμοποιούν και όλοι οι πελάτες του.

Skill: Intermediate
Tags: Git, Git Hooks, Hugo

Αυτόματο deployment Hugo site μέσω Git Hooks

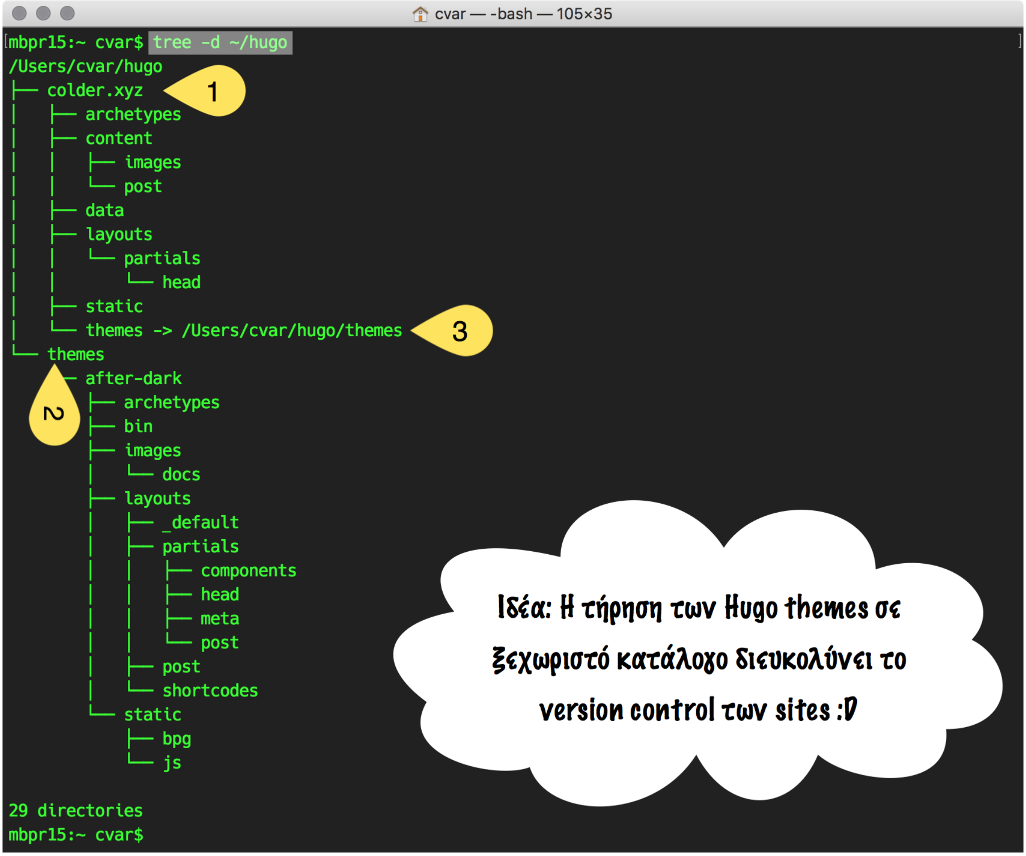
Σε προηγούμενο άρθρο γνωρίσαμε το Hugo, έναν δημοφιλή και ταχύτατο static site generator. Το δουλέψαμε και μάθαμε πώς να το χρησιμοποιούμε ώστε, ξεκινώντας από το μηδέν, να δημιουργούμε σύγχρονα blogs ή/και ελκυστικά project sites. Σ' αυτό το άρθρο θα δείξουμε, αναλυτικά και βήμα προς βήμα, πώς υποβάλλουμε ένα Hugo site υπό version control με τη βοήθεια του Git, καθώς και πώς αναπτύσσουμε (deploy) νέο περιεχόμενο με τη βοήθεια του προαναφερθέντος Version Control System και των λεγόμενων Git Hooks.

Για τη συνέχεια υποθέτουμε ότι έχετε ήδη φτιάξει το πρώτο σας static site. Αυτό με το οποίο εμείς δουλέψαμε για τις ανάγκες του παρόντος, ξεκίνησε την πορεία του από τον τοπικό κατάλογο ~/colder-dot-xyz ενός Mac Book και, παρεμπιπτόντως, είναι δημοσιευμένο στο https://colder.xyz. Περιττό να σημειώσουμε ότι δεν χρειάζεται να δουλέψετε κι εσείς σε περιβάλλον Mac OS.

Πριν υποβάλουμε το site μας υπό version control, φτιάξαμε έναν νέο κατάλογο, τον ~/hugo, μεταφέραμε εκεί τον ~/colder-dot-xyz και τον μετονομάσαμε σε colder.xyz. Βεβαίως, η δημιουργία του ~/hugo δεν έγινε μόνο και μόνο για να μετακινήσουμε (και να μετονομάσουμε) τον ~/colder-dot-xyz. Εντός του ~/hugo φτιάξαμε και τον κατάλογο themes, μεταβήκαμε σ' αυτόν και κατεβάσαμε την πλέον πρόσφατη έκδοση του theme του site μας. Αμέσως μετά διαγράψαμε τον παλιό κατάλογο ~/hugo/colder.xyz/themes και τον αντικαταστήσαμε με ένα ομώνυμο symbolic link προς τον ~/hugo/themes. Γιατί όλες αυτές οι διευθετήσεις; Κατ' αρχάς, η παρουσία του Git repository με το theme θα περιέπλεκε το version control του site. Κατά δεύτερον, όταν έχουμε περισσότερα από ένα Hugo sites είναι πολύ πιο βολικό να διατηρούμε τα themes σε έναν ξεχωριστό κατάλογο κι από τους καταλόγους των sites να δημιουργούμε links προς αυτόν.

Μετατροπή του site σε Git repository

Πριν πιάσουμε δουλειά, να πούμε ότι αν τώρα αρχίζετε με το Git τότε ίσως είναι καλή ιδέα να διαβάσετε το σχετικό εισαγωγικό άρθρο που έχουμε δημοσιεύσει (https://deltahacker.gr/?p=16654).



Στον κατάλογο ~/hugo μεταφέραμε εκείνον του site μας (1) και φτιάξαμε κι άλλον έναν, για την τήρηση των Hugo themes (2). Από τον ~/hugo/colder.xyz διαγράψαμε τον κατάλογο themes και στην θέση του δημιουργήσαμε ένα symbolic link προς τον ~/hugo/themes (3). Με αυτή την πολύ απλή διευθέτηση διευκολύνεται το version control του colder.xyz. Όταν εξάλλου αναπτύσσουμε/συντηρούμε περισσότερα από ένα Hugo sites αποφεύγουμε και τα περιττά downloads, αφού τα themes παραμένουν εκτός των καταλόγων των sites.

Για τη μετατροπή του site σε Git repository, απλά μεταβαίνουμε στον κατάλογό του (π.χ., cd ~/hugo/colder.xyz) και πληκτρολογούμε:

```
$ git init
```

Εξ ορισμού, το Git αφήνει τα κενά directories εκτός των commits. Μερικές φορές όμως το Hugo τα χρειάζεται, οπότε πρέπει τώρα να πείσουμε το Git ν' αλλάξει συνήθειες. Προς τούτο, αρκεί να τοποθετήσουμε ένα κρυμμένο αρχείο με όνομα .gitkeep στη ρίζα **κάθε** καταλόγου αμέσως κάτω από εκείνον του site μας — **εκτός** από τον .git:

```
$ for each_dir in $(find . -type d -depth 1 -not -name ".*"); \
```

```
> do touch "$each_dir"/.gitkeep; done
```

Επειδή εξάλλου το περιεχόμενο του site πρέπει να παράγεται από την αρχή **πριν από κάθε deployment**, θα αφήσουμε τον κατάλογο public εκτός version control:

```
$ echo "public" >> .gitignore
```

Αν, παρεμπιπτόντως, δουλεύετε κι εσείς σε περιβάλλον Mac OS, στο αρχείο .gitignore προσθέστε και μια γραμμή γι' αυτά τα ευλογημένα αρχεία .DS_Store:

```
$ echo ".DS_Store" >> .gitignore
```

Προσθέτουμε τώρα στο Git όλα τα αρχεία και τους καταλόγους εντός του colder.xyz που είναι **untracked**:

```
$ git add .
```

Επαληθεύουμε ότι όλα είναι όπως πρέπει:

```
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   archetypes/.gitkeep
        new file:   archetypes/default.md
        new file:   config.toml
        new file:   content/.gitkeep
        new file:   content/about.md
        new file:   content/images/reykjavik_fog.jpg
        new file:   content/post/do-you-speak-icelandic.md
        new file:   content/post/drivels-and-delusions.md
        new file:   content/post/elcap.md
        new file:   content/post/exploding-disks.md
        new file:   content/post/from-ios-to-android.md
        new file:   content/post/heatwave-and-fog.md
```

```
        new file:   content/post/ice-drone-homage.md
        new file:   content/post/kjotsupudagurinn.md
        new file:   content/post/no-more-doc.md
        new file:   content/post/old-pcs-selfheal.md
        new file:   content/post/sierraflow.md
        new file:   content/post/static-site-gen-at-random.md
        new file:   content/post/stubborn-weather.md
        new file:   data/.gitkeep
        new file:   layouts/.gitkeep
        new file:   layouts/partials/head/favicon.html
        new file:   static/.gitkeep
        new file:   themes
```

Ολοκληρώνουμε πραγματοποιώντας το πρώτο commit για το Hugo site μας:

```
$ git commit -m "Initial commit, some content already exists"
[master (root-commit) 8e27a93] Initial commit, some content already exists
 25 files changed, 460 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 archetypes/.gitkeep
 create mode 100644 archetypes/default.md
 create mode 100644 config.toml
 create mode 100644 content/.gitkeep
 create mode 100644 content/about.md
 create mode 100644 content/images/reykjavik_fog.jpg
 create mode 100644 content/post/do-you-speak-icelandic.md
 create mode 100644 content/post/drivels-and-delusions.md
 create mode 100644 content/post/elcap.md
 create mode 100644 content/post/exploding-disks.md
 create mode 100644 content/post/from-ios-to-android.md
 create mode 100644 content/post/heatwave-and-fog.md
 create mode 100644 content/post/ice-drone-homage.md
 create mode 100644 content/post/kjotsupudagurinn.md
 create mode 100644 content/post/no-more-doc.md
 create mode 100644 content/post/old-pcs-selfheal.md
 create mode 100644 content/post/sierraflow.md
 create mode 100644 content/post/static-site-gen-at-random.md
```

```
create mode 100644 content/post/stubborn-weather.md
create mode 100644 data/.gitkeep
create mode 100644 layouts/.gitkeep
create mode 100644 layouts/partials/head/favicon.html
create mode 100644 static/.gitkeep
create mode 120000 themes
```

Πάρα πολύ ωραία, είμαστε έτοιμοι για τη συνέχεια. Στο εξής, το μηχάνημα από το οποίο εργαζόμαστε θα το ονομάζουμε **development workstation** ή απλά **development**. Εκείνο όπου θα καταλήγει το περιεχόμενό μας –μ' άλλα λόγια εκείνο στο οποίο τρέχει ο web server που σερβίρει το site μας–, θα το ονομάζουμε **production server** ή απλά **production**. Από το development οφείλουμε να έχουμε πρόσβαση στο production μέσω SSH και μάλιστα χωρίς την πληκτρολόγηση password. (Αν χρειαστεί διαβάστε το σχετικό άρθρο περί SSH password-less logins, στο https://deltahacker.gr/?p=13357.)

Μεταφορά αρχικού Git repository στο production

Στο development workstation μεταβαίνουμε στον κατάλογο ακριβώς πάνω από τον colder.xyz...

```
$ pwd
/Users/cvar/hugo/colder.xyz
$ cd ..
$ pwd
/Users/cvar/hugo
```

...κι εντός του προσωπικού καταλόγου του χρήστη μας (cvar) δημιουργούμε ένα **bare Git repository** πληκτρολογώντας κάτι σαν το ακόλουθο:

```
$ git clone --bare colder.xyz ~/colder.xyz.git
Cloning into bare repository '/Users/cvar/colder.xyz.git'...
done.
```

Τι είναι όμως αυτό το bare Git repository; Πολύ απλά, είναι ένας κατάλογος (εν προκειμένω με όνομα colder.xyz.git) που αντί να έχει όλα τα αρχεία διαχείρισης του αρχικού Git repository μέσα στον κρυφό υποκατάλογο .git, τα έχει στη ρίζα του:

```
$ ls -lh ~/colder.xyz.git/
total 32
```

```
-rw-r--r--   1 cvar  staff    23B 14 jan 12:32 HEAD
-rw-r--r--   1 cvar  staff   164B 14 jan 12:32 config
-rw-r--r--   1 cvar  staff    73B 14 jan 12:32 description
drwxr-xr-x  12 cvar  staff   384B 14 jan 12:32 hooks
drwxr-xr-x   3 cvar  staff    96B 14 jan 12:32 info
drwxr-xr-x  38 cvar  staff   1,2K 14 jan 12:32 objects
-rw-r--r--   1 cvar  staff   105B 14 jan 12:32 packed-refs
drwxr-xr-x   4 cvar  staff   128B 14 jan 12:32 refs
```

Τον κατάλογο αυτόν τον μεταφέρουμε στον production server, ώστε σε λίγο να είμαστε σε θέση να στέλνουμε (push) εύκολα περιεχόμενο εκεί:

```
$ scp -r ~/colder.xyz.git sub0@box.colder.xyz:~/
```

Το hostname του δικού μας production είναι box.colder.xyz κι ο απομακρυσμένος μας χρήστης εκεί έχει username το sub0. Προφανώς, για τη δική σας περίπτωση θα κάνετε τις απαραίτητες προσαρμογές.

Το production ως Git remote

Στο development, πάμε ξανά στον κατάλογο του Hugo site (cd ~/hugo/colder.xyz) και γράφουμε:

```
$ git remote add colder-prod sub0@colder.xyz:colder.xyz.git
```

Μόλις προσθέσαμε ένα νέο remote για το τοπικό Git repository. Βρίσκεται στο production κι επιλέξαμε να το ονομάσουμε colder-proc:

```
$ git remote -v
colder-prod sub0@colder.xyz:colder.xyz.git (fetch)
colder-prod sub0@colder.xyz:colder.xyz.git (push)
```

Πριν κάνουμε οτιδήποτε με το remote, οφείλουμε πρώτα να τακτοποιήσουμε συγκεκριμένα θέματα στο production. Αναλυτικότερα, το απομακρυσμένο μηχάνημα χρειάζεται να έχει εγκατεστημένα τα Git, Hugo, Pygments – και φυσικά έναν web server. Συνδεθείτε εκεί μέσω SSH και, αναλόγως της διανομής Linux, πράξτε τα δέοντα ώστε να εγκαταστήσετε τα αντίστοιχα πακέτα λογισμικού. Πληροφοριακά, το δικό μας production τρέχει Ubuntu 14.04.5 LTS. Ως web server είχε ήδη τον nginx, οπότε δεν χρειάστηκε να στραφούμε σε κάποιον άλλον. Τα Git και Pygments τα εγκαταστήσαμε κατά τα γνωστά, δηλαδή με ένα sudo apt install git python-pygments.

Για την πλέον πρόσφατη έκδοση του Hugo, η οποία τη στιγμή που γραφόταν το παρόν ήταν η 0.32.4, κατεβάσαμε το αντίστοιχο 64μπιτο Debian package από το https://github.com/gohugoio/hugo/releases...

```
$ wget https://github.com/gohugoio/hugo/releases/download/v0.32.4/hugo_0.32.4_
Linux-64bit.deb
```

...και το εγκαταστήσαμε ως εξής:

```
$ sudo dpkg -i hugo_0.32.4_Linux-64bit.deb
Selecting previously unselected package hugo.
(Reading database ... 142359 files and directories currently installed.)
Preparing to unpack hugo_0.32.4_Linux-64bit.deb ...
Unpacking hugo (0.32.4) ...
Setting up hugo (0.32.4) ...
$ hugo version
Hugo Static Site Generator v0.32.4 linux/amd64 BuildDate: 2018-01-
11T10:58:37+02:00
```

Μην ξεχνάτε ότι στο production χρειαζόμαστε και το theme ή τα themes για τα static site μας. Αν δεν έχει γίνει ακόμα προφανές, ας σημειώσουμε ότι από το development θα προωθούμε περιεχόμενο στο production, αυτό θα κτίζεται αυτόματα εκεί κι αμέσως μετά θα διατίθεται δημόσια μέσω του web server. Στον προσωπικό κατάλογο του απομακρυσμένου μας χρήστη, λοιπόν, φτιάχνουμε έναν νέο ονόματι themes. Εκεί μέσα κλωνοποιούμε όποια κι όσα themes θέλουμε. Παράδειγμα:

```
$ mkdir ~/themes && cd themes
$ git clone https://github.com/comfusion/after-dark.git
Cloning into 'after-dark'...
remote: Counting objects: 2211, done.
remote: Total 2211 (delta 0), reused 0 (delta 0), pack-reused 2211
Receiving objects: 100% (2211/2211), 17.21 MiB | 7.82 MiB/s, done.
Resolving deltas: 100% (1212/1212), done.
Checking connectivity... done.
$ git clone https://github.com/calintat/minimal.git
Cloning into 'minimal'...
remote: Counting objects: 350, done.
remote: Compressing objects: 100% (7/7), done.
Receiving objects: 100% (350/350), 308.68 KiB | 0 bytes/s, done.
```

```
remote: Total 350 (delta 4), reused 6 (delta 3), pack-reused 340
Resolving deltas: 100% (176/176), done.
Checking connectivity... done.
```

Έχουμε αρκετή ακόμη δουλειά στο production. Στο σημείο αυτό όμως αξίζει να κάνουμε ένα σύντομο διάλειμμα κι από το development να τσεκάρουμε αν (και πώς) φαίνεται το remote που προσθέσαμε πριν λίγο. Από το development, λοιπόν, και συγκεκριμένα μέσα από τον κατάλογο του site μας, αρκεί να πληκτρολογήσουμε κάτι τέτοιο:

```
$ cd ~/hugo/colder.xyz
$ git ls-remote colder-prod
94cd450f0ff7d7c6bdb740d18cb05bf24ad620f7   HEAD
94cd450f0ff7d7c6bdb740d18cb05bf24ad620f7   refs/heads/master
```

Παρατηρώντας το αποτέλεσμα της δεύτερης εντολής, γίνεται φανερό ότι το Git μπορεί κι εγκαθιδρύει συνδέσεις μεταξύ development και production – κι αυτό είναι πολύ καλό κι ακριβώς ό,τι θέλουμε.

Αυτόματη επεξεργασία στο production

Κάθε φορά που από το development κάνουμε ένα git push περιεχομένου στο production, θέλουμε να υπάρχει εκεί ένα script (το post-receive) που θα εκτελείται αυτόματα και θα αναλαμβάνει το κτίσιμο του site και φυσικά τη δημοσίευση του νέου περιεχομένου. Συνδεόμαστε, λοιπόν, στον production server και μεταβαίνουμε στον κατάλογο ονόματι hooks, εντός του bare Git repository που στείλαμε από το development workstation προηγουμένως:

```
$ cd colder.xyz.git/hooks/
$ ls -lh
total 44K
-rwxr-xr-x 1 sub0 sub0  478 Jan 14 13:51 applypatch-msg.sample
-rwxr-xr-x 1 sub0 sub0  896 Jan 14 13:51 commit-msg.sample
-rwxr-xr-x 1 sub0 sub0  189 Jan 14 13:51 post-update.sample
-rwxr-xr-x 1 sub0 sub0  424 Jan 14 13:51 pre-applypatch.sample
-rwxr-xr-x 1 sub0 sub0 1.7K Jan 14 13:51 pre-commit.sample
-rwxr-xr-x 1 sub0 sub0 1.5K Jan 14 13:51 prepare-commit-msg.sample
-rwxr-xr-x 1 sub0 sub0 1.4K Jan 14 13:51 pre-push.sample
-rwxr-xr-x 1 sub0 sub0 4.8K Jan 14 13:51 pre-rebase.sample
-rwxr-xr-x 1 sub0 sub0  544 Jan 14 13:51 pre-receive.sample
```

```
-rwxr-xr-x 1 sub0 sub0 3.6K Jan 14 13:51 update.sample
```

Όπως διαπιστώνετε, υπάρχουν αρκετά δείγματα scripts. Ρίξτε, αν θέλετε, μια ματιά σε κάποια εξ αυτών. Ας φτιάξουμε τώρα ένα νέο, με όνομα post-receive και χωρίς κατάληξη:

```
$ touch post-receive
```

Το post-receive θα είναι, ουσιαστικά, ένα BASH script, το οποίο θα εκτελείται αυτόματα κάθε φορά που θα στέλνουμε (push) νέο περιεχόμενο στο production. Ανοίγουμε το νέο αρχείο με τον text editor της προτίμησής μας (nano, vim κ.ο.κ.) και φροντίζουμε ώστε να έχει περιεχόμενο σαν το ακόλουθο:

```
#!/bin/bash
SITE_REPO="$HOME/colder.xyz.git"
WORKING="$HOME/colder.xyz-working"
PUBLIC="/home/user-data/www/default"
BACKUP="$HOME/colder.xyz-backup"
DOMAIN="colder.xyz"
PROTO="https"
HUGO="/usr/local/bin/hugo"
rm -rf "$WORKING"
rsync -aqz "$PUBLIC"/ "$BACKUP"
rsync_ec="$?"
if [ $rsync_ec -ne 0 ]; then
      printf "failed to create a backup of public directory, "
      printf "rsync returned error code %d\n" $rsync_ec
      rm -rf "$BACKUP"
      exit 1
fi
git clone "$SITE_REPO" "$WORKING"
git_ec="$?"
if [ $git_ec -ne 0 ]; then
      printf "failed to git-clone site repository into working "
      printf "directory, git returned error code %d\n" $git_ec
      rm -rf "$WORKING"
      exit 1
fi
```

```
rm -rf "$PUBLIC"/*
"$HUGO" --source "$WORKING" --destination "$PUBLIC" \
      --baseURL "$PROTO://$DOMAIN/"
hugo_ec="$?"
if [ $hugo_ec -ne 0 ]; then
      printf "failed to properly deploy site, "
      printf "hugo returned error code %d\n" $hugo_ec
      rm -rf "$PUBLIC"/*
      rsync -aqz "$BACKUP"/ "$PUBLIC"
fi
rm -rf "$WORKING"
```

Νομίζουμε ότι το script είναι αρκετά human readable και η λειτουργία του είναι μάλλον προφανής. Προσέξτε μόνο ώστε να τροποποιήσετε τις μεταβλητές στην αρχή, ώστε να αντικατοπτρίζουν το δικό σας περιβάλλον. Δώστε λίγη παραπάνω προσοχή στην PUBLIC, η οποία κρατά την πλήρη διαδρομή του καταλόγου από τον οποίο ο web server σερβίρει το site. Στο δικό μας setup ο nginx ήταν ήδη ρυθμισμένος ώστε να χρησιμοποιεί το /home/user-data/www/default – και δεν είχαμε λόγο για κάποια αλλαγή. Στο δικό σας setup είναι πολύ πιθανό να τροποποιήσετε κατάλληλα το configuration του web server ή/και την τιμή της PUBLIC. Από τη στιγμή βέβαια που έχετε ήδη φτιάξει το δικό σας static site, το πιθανότερο είναι ότι θα πειράξετε μόνο τη μεταβλητή. Προσέξτε, παρακαλούμε, και τη μεταβλητή WORKING, η οποία δέχεται το full path ενός καταλόγου ο οποίος αποτελεί κλώνο του bare Git repository. Απ' αυτόν τον κατάλογο το Hugo θα κτίζει το site μας, οπότε αυτός θα πρέπει να βρίσκεται στο ίδιο επίπεδο με τον κατάλογο themes.

Το αρχείο (script) post-receive οφείλουμε να το κάνουμε εκτελέσιμο:

```
$ chmod +x post-receive
```

Ώρα για δοκιμές

Στο development workstation μεταβαίνουμε στον κατάλογο του Hugo site (cd ~/hugo/colder.xyz) και φτιάχνουμε ένα νέο, δοκιμαστικό post:

```
$ hugo new post/siglufjordur.md
/Users/cvar/hugo/colder.xyz/content/post/siglufjordur.md created
```

Δεν είναι κακή ιδέα να προσθέσουμε λίγο περιεχόμενο, ίσως να ενθέσουμε και μια φωτογραφία. Δείτε, π.χ., πώς μοιάζει το δικό μας δοκιμαστικό αρχείο:

```
$ cat content/post/siglufjordur.md
---
title: "Δοκιμαστικό για ανάπτυξη Hugo site μέσω Git Hooks"
date: 2018-01-15T11:47:28+01:00
tags: [ "Hugo", "Git" ]
draft: true
---
```

Όπως και να το κάνουμε, αλλιώς είναι να δημοσιεύεις περιεχόμενο μεταφέροντας αρχεία μέσω rsync ή scp, κι αλλιώς με ένα git push :) Μ' αυτή την προσέγγιση υπάρχουν πολλά καλά. Πας, π.χ., σε ένα οποιοδήποτε development machine που έχει Git και μπορείς να κλωνοποιήσεις το static site σου με ένα git clone. Ή αν το έχεις ήδη κλωνοποιήσει, πληκτρολογείς git pull ώστε να είσαι up-to-date. Αργότερα προσθέτεις νέο περιεχόμενο ή/και τροποποιείς το υπάρχον, ενημερώνεις το production δίνοντας git push, αισθάνεσαι μεγάλη άνεση, χαρά, καθώς κι ότι έχεις τον έλεγχο.

Δικαίως.

Κι ας κλείσουμε αυτό το δοκιμαστικό post με μια φωτογραφία από ένα πολύχρωμο (και πρωτότυπο) μουσείο στο Siglufjörður.

```
![Grána, Siglufjörður | Πηγή: http://rooksack.de/siglufjoerdur-schoenste-stadt-island]
(/images/siglufjordur.jpg "Grána, Siglufjörður | Πηγή: http://rooksack.de/siglufjoerdur-schoenste-stadt-island")
_The Grána fish meal and oil factory is a unique exhibition about an important chapter in the economic and social history of Siglufjörður and Iceland. (Visit [The Herring Era Museum](http://www.sild.is/en/exhibitions/grana/) for more.)_
```

Έχουμε τοποθετήσει κι ένα αρχείο εικόνας στον κατάλογο content/images, οπότε τώρα η κατάσταση του τοπικού Git repository έχει ως εξής:

```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        content/images/siglufjordur.jpg
        content/post/siglufjordur.md
nothing added to commit but untracked files present (use "git add" to track)
```

Ας αρχίσουμε να παρακολουθούμε τα δύο νέα αρχεία, δηλαδή το siglufjordur.md στον κατάλογο content/post, καθώς και το siglufjordur.jpg στον κατάλογο content/images:

```
$ git add .
$ git status
On branch master
Changes to be committed:
 (use "git reset HEAD <file>..." to unstage)
      new file:   content/images/siglufjordur.jpg
      new file:   content/post/siglufjordur.md
```

Προς το παρόν δεν θα κάνουμε κάποια άλλη αλλαγή/προσθήκη στο δοκιμαστικό post, οπότε έχει έλθει η ώρα για commit. Απλά, πριν το κάνουμε φροντίζουμε ώστε στο siglufjordur.md η μεταβλητή draft να είναι false (προς το παρόν είναι true, κι αυτό σημαίνει ότι το Hugo δεν θα δημοσιεύσει το post). Ιδού το commit, λοιπόν:

```
$ git commit -m "Add new files for testing Git hooks deploy" .
[master 71684f6] Add new files for testing Git hooks deploy
 2 files changed, 16 insertions(+)
 create mode 100644 content/images/siglufjordur.jpg create mode 100644 content/
post/siglufjordur.md
```

Στη συνέχεια θα προωθήσουμε (push) το νέο περιεχόμενο στον απομακρυσμένο production server. Γράφουμε:

```
$ git push colder-prod master
Counting objects: 5, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 439 bytes | 439.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
remote: Cloning into '/home/sub0/colder.xyz-working'...
remote: done.
remote:
remote:                    | EN
remote: +------------------+-----+
remote:   Pages            | 104
remote:   Paginator pages  |   9
remote:   Non-page files   |   2
remote:   Static files     |   2
remote:   Processed images |   0
```

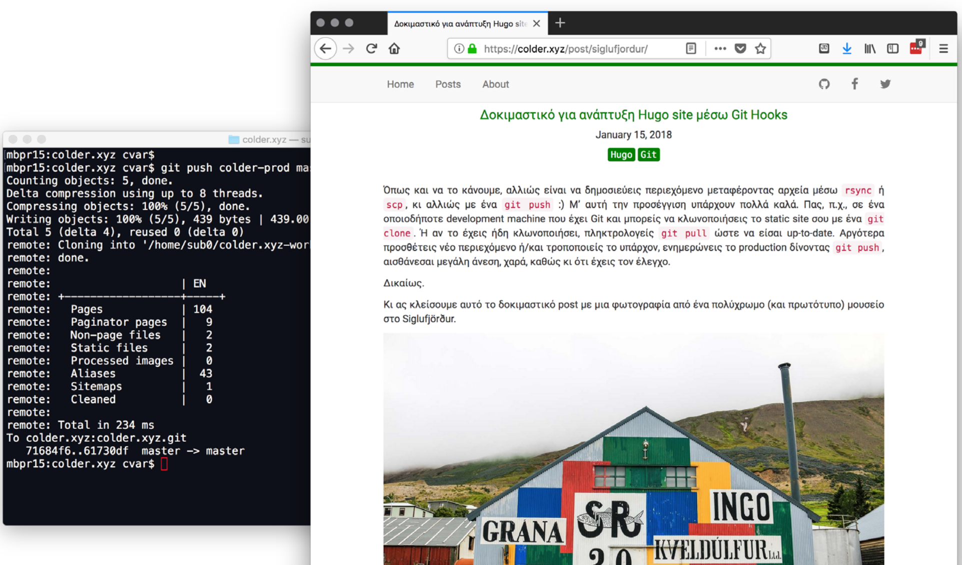
```
remote:   Aliases             |  43
remote:   Sitemaps            |   1
remote:   Cleaned             |   0
remote:
remote: Total in 234 ms
To colder.xyz:colder.xyz.git
   71684f6..61730df  master -> master
```

Δεν φαίνεται να υπάρχει κάποιο πρόβλημα, μάλλον όλα δούλεψαν κατ' ευχήν. Φυσικά, προκειμένου να είμαστε 101% σίγουροι, από τον web browser της προτίμησής μας επισκεπτόμαστε το site μας και ψάχνουμε για το νέο post.

Και μια σημείωση: Αν κάθε φορά που κάνετε push δεν θέλετε να πληκτρολογείτε ούτε το όνομα του remote (π.χ., colder-prod) κι ούτε το τρέχον τοπικό branch (master), πληκτρολογήστε:

```
$ git push --set-upstream colder-prod master
Branch 'master' set up to track remote branch 'master' from 'colder-prod'.
```

Βεβαίως, στη θέση του colder-prod εσείς θα βάλετε το όνομα του δικού σας remote.



Αυτόματο deployment περιεχομένου στο production μέσω Git Hooks – και θα θέλετε να ξεχάσετε οποιονδήποτε άλλη μέθοδο εφαρμόζατε έως σήμερα. Τώρα που το σκεφτόμαστε, το πιθανότερο είναι να θέλετε να ξεχάσετε την παλιά μέθοδο.

Εργασία από άλλο development workstation

Πάμε τώρα σε κάποιο άλλο μηχάνημα, αυτή τη φορά σε ένα που τρέχει Linux κι έχει εγκατεστημένα τα Git, Hugo και Pygments. Είμαστε συνεπώς σε θέση να αναπτύσσουμε και σ' αυτό το workstation περιεχόμενο για το static site μας. Προς το παρόν δεν έχουμε κανένα αρχείο του site στο νέο development, αλλά αυτό διορθώνεται εύκολα. Πράγματι, αρκεί να μεταβούμε σε έναν βολικό κατάλογο (π.χ., στον ~/hugo) και να κλωνοποιήσουμε το κατάλληλο repository από τον απομακρυσμένο production server:

```
> git clone sub0@box.colder.xyz:~/colder.xyz.git
Cloning into 'colder.xyz'...
remote: Counting objects: 87, done.
remote: Compressing objects: 100% (80/80), done.
remote: Total 87 (delta 39), reused 0 (delta 0)
Receiving objects: 100% (87/87), 582.73 KiB | 1.54 MiB/s, done.
Resolving deltas: 100% (39/39), done.
> ls -lh
total 4,0K
drwxr-xr-x 8 userson users 4,0K jan 15 14:12 colder.xyz
> ls -lh colder.xyz/
total 24K
drwxr-xr-x 2 userson users 4,0K jan 15 14:12 archetypes
-rw-r--r-- 1 userson users  924 jan 15 14:12 config.toml
drwxr-xr-x 4 userson users 4,0K jan 15 14:12 content
drwxr-xr-x 2 userson users 4,0K jan 15 14:12 data
drwxr-xr-x 3 userson users 4,0K jan 15 14:12 layouts
drwxr-xr-x 2 userson users 4,0K jan 15 14:12 static
lrwxrwxrwx 1 userson users    9 jan 15 14:12 themes -> ../themes
```

Αυτό το symbolic link, μέσα στον κατάλογο ~/hugo/colder.xyz, ίσως δείχνει σε κατάλογο που δεν υπάρχει. Σε μια τέτοια περίπτωση, απλά φτιάχνουμε τον κατάλογο themes στο parent directory κι εντός αυτού κατεβάζουμε τα Hugo themes που θέλουμε.

Απ' αυτή τη στιγμή και στο εξής, για το static site μας μπορούμε να δουλεύουμε και από το development workstation με το Linux. Κάθε φορά που θα έχουμε νέο περιεχόμενο για τον production server, θα το προωθούμε εκεί με ένα απλό git push. Τέλος, από ένα οποιοδήποτε άλλο development θα ενημερώνουμε το τοπικό repository του site με ένα git pull.

Χαρείτε τον νέο, μοντέρνο κι εξαιρετικά διασκεδαστικό τρόπο αυτόματης ανάπτυξης του περιεχομένου σας.

Skill: Beginner
Tags: PXE, PXE server, SystemRescueCd

Εφαρμογή PXE boot: Εκκίνηση SystemRescueCd

Ένας PXE server διευκολύνει σημαντικά τις εγκαταστάσεις λειτουργικών συστημάτων σε φυσικούς ή εικονικούς hosts του τοπικού δικτύου. Βρίσκει όμως κι άλλες εφαρμογές. Μία απ' αυτές αφορά στην απομακρυσμένη εκκίνηση ενός OS άμεσης ανάγκης, για υπολογιστές που είτε δεν είναι σε θέση να φορτώσουν το δικό τους είτε χρειάζεται να τους κάνουμε επεμβάσεις που δεν είναι εφικτές μέσα από το ήδη εγκατεστημένο OS.

Μια διανομή Linux για την επιδιόρθωση/διαχείριση ενός συστήματος ή/και για τη λήψη backup μετά από κάποιο σοβαρό crash, είναι το SystemRescueCd (http://www.system-rescue-cd.org). Η εν λόγω διανομή δεν είναι χρήσιμη μόνο για υπολογιστές με Linux. Για παράδειγμα, το εργαλείο GParted που περιλαμβάνεται στην εργαλειοθήκη της είναι ικανό **και** για τη διαχείριση κατατμήσεων με Windows filesystems. Το SystemRescueCd ανήκει στην κατηγορία των Live distributions, που σημαίνει ότι ξεκινά από CD/DVD ή USB (thumb) drive, τρέχει από τη μνήμη και δεν χρειάζεται εγκατάσταση. Στη συνέχεια θα δείξουμε πώς φορτώνουμε το SystemRescueCd σε κάποιον υπολογιστή **δικτυακά**, από έναν **PXE server**. Πιο συγκεκριμένα, υποθέτουμε ότι στο Raspberry Pi σας λειτουργεί ήδη PXE server, με βάση τις οδηγίες στο σχετικό μας άρθρο στο https://deltahacker.gr/?p=17693.

Ξεκινάμε κατεβάζοντας το ISO image του SystemRescueCd 5.1.1 από τη σελίδα download του project: http://www.system-rescue-cd.org/Download. Εμείς, το αρχείο που πήραμε μετονομάσαμε σε SystemRescueCD-x86-5.1.1.iso και τ' αποθηκεύσαμε στον κατάλογο /mnt/external/ISOs. Το περιεχόμενο του image το προσαρτήσαμε κάτω από το /var/www/html/distributions/systemrescuecd-x86-5.1.1, ώστε να είναι προσβάσιμο από τον web server που τρέχει στο Raspberry (είναι ο lighttpd). Προσθέσαμε κι αυτή τη γραμμή στο /etc/fstab…

```
/mnt/external/ISOs/SystemRescueCD-x86-5.1.1.iso
      /var/www/html/distributions/systemrescuecd-x86-5.1.1
         iso9660 defaults,ro 0 0
```

(είναι **μία γραμμή** κι όχι τρεις), ώστε η προσάρτηση να γίνεται αυτόματα κατά την εκκίνηση του Raspbian.

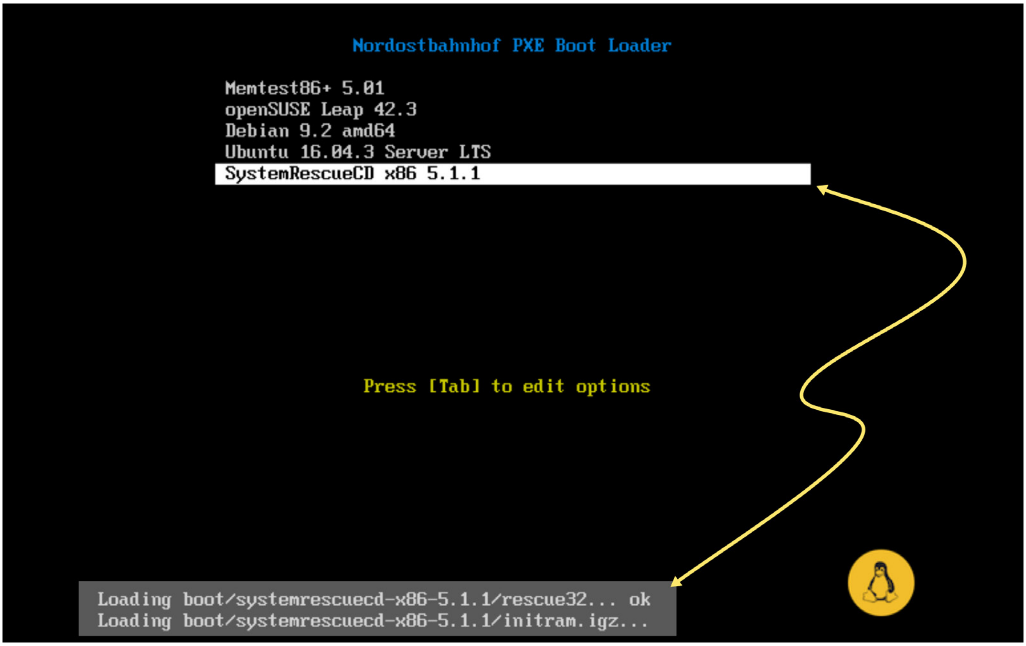
Στη συνέχεια ενημερώνουμε τον PXE server για τα αρχεία του πυρήνα και του initial ramdisk image, τα οποία θα φορτώνουν στους PXE clients όποτε αυτοί ζητούν την εκκίνηση του SystemRescueCd. Αναλυτικότερα, από τον κατάλογο /var/www/html/distributions/systemrescuecd-x86-5.1.1/isolinux αντιγράψαμε τα αρχεία rescue32 (πυρήνας) και initram.igz (initial ramdisk image) στον κατάλογο /mnt/external/tftpboot/boot/systemrescuecd-x86-5.1.1 (ο systemrescuecd-x86-5.1.1 δεν υπήρχε κάτω από τον /mnt/external/tftpboot/boot, γι' αυτό και πρώτα τον φτιάξαμε). Το rescue32 φορτώνει σε συστήματα αρχιτεκτονικής 32bit ή 64bit. Υπάρχει και το rescue64, το οποίο φορτώνει μόνο σε συστήματα 64bit. Στο δικό μας setup χρησιμοποιούμε το rescue32 — και το έχουμε δοκιμάσει επιτυχώς **και** σε 64bit hosts.

Σειρά έχει η δημιουργία ενός νέου entry για το μενού που προσφέρει ο PXE server στους clients. Στο αρχείο default, μέσα στον κατάλογο /mnt/external/tftpboot/pxelinux.cfg, προσθέσαμε τις ακόλουθες γραμμές:

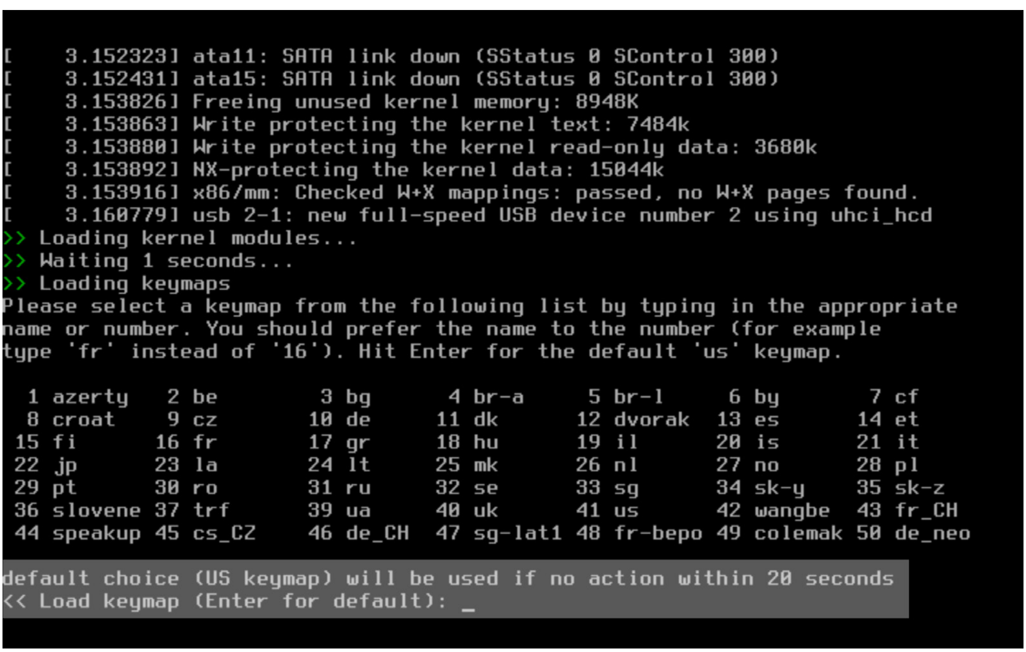
```
LABEL SystemRescueCD-x86-5.1.1
   MENU LABEL SystemRescueCD x86 5.1.1
   KERNEL boot/systemrescuecd-x86-5.1.1/rescue32
   APPEND
          initrd=
           boot/systemrescuecd-x86-5.1.1/initram.igz
          netboot=
           http://192.168.178.2/distributions/systemrescuecd-x86-5.1.1/sysrcd.dat
```

(οι πέντε γραμμές από την APPEND έως και την τελευταία, **στην πραγματικότητα είναι μία**). Βεβαίως, στο δικό σας default θα βάλετε τη διεύθυνση IP του PXE server σας — κι εννοείται ότι θα τροποποιήσετε paths κι ονόματα καταλόγων.

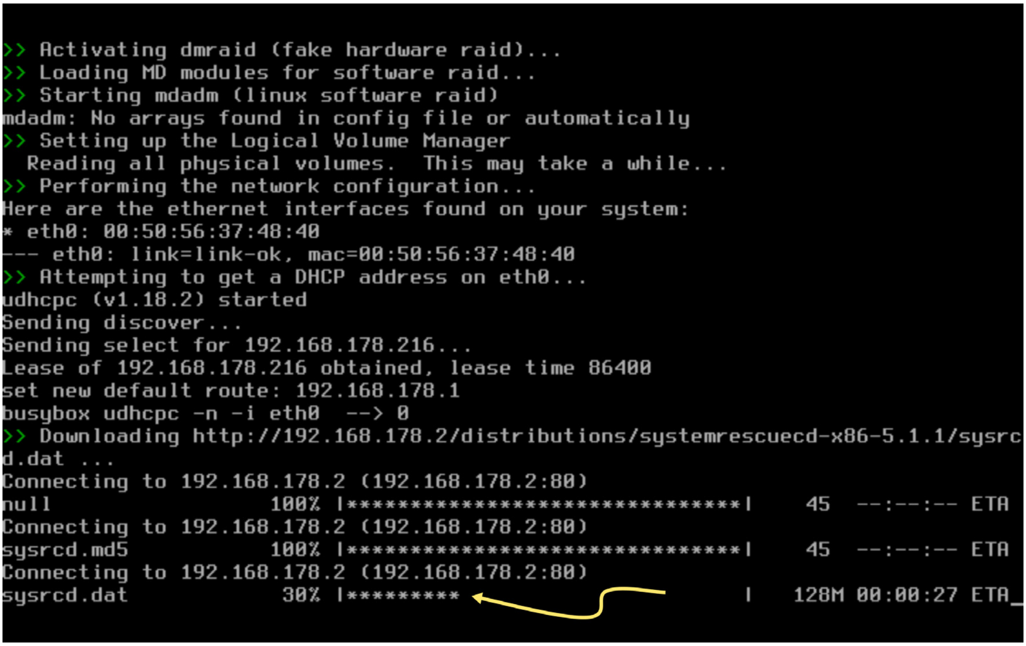
Μετά τα παραπάνω είμαστε έτοιμοι να δοκιμάσουμε αν το SystemRescueCd φορτώνει απομακρυσμένα στους PXE clients. Αρκεί να (επαν)εκκινήσουμε έναν φυσικό ή εικονικό host, φροντίζοντας ώστε στο boot sequence πρώτη συσκευή να είναι η κάρτα δικτύου. Δείτε τα screenshots που ακολουθούν, διαβάστε και τις αντίστοιχες περιγραφές.



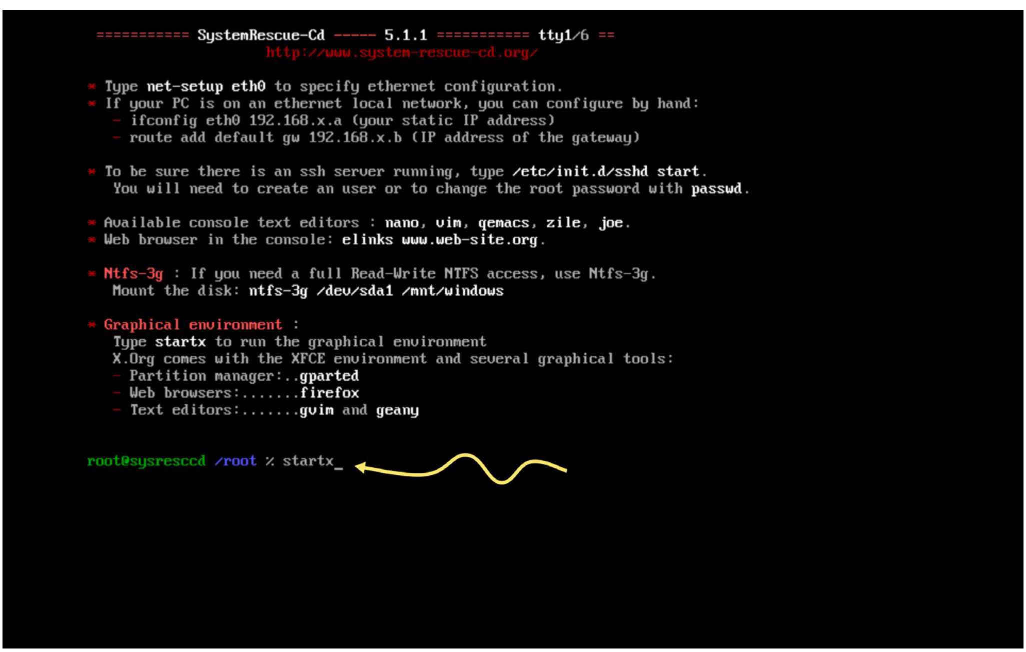
Νέα επιλογή στο μενού του PXE server που φιλοξενεί το Raspberry Pi μας: Πρόκειται για το SystemRescueCd, ένα σύστημα για επεμβάσεις ή/και backups σε συστήματα που είτε δεν έχουν λειτουργικό είτε έχουν υποστεί κάποιο καταστροφικό crash.



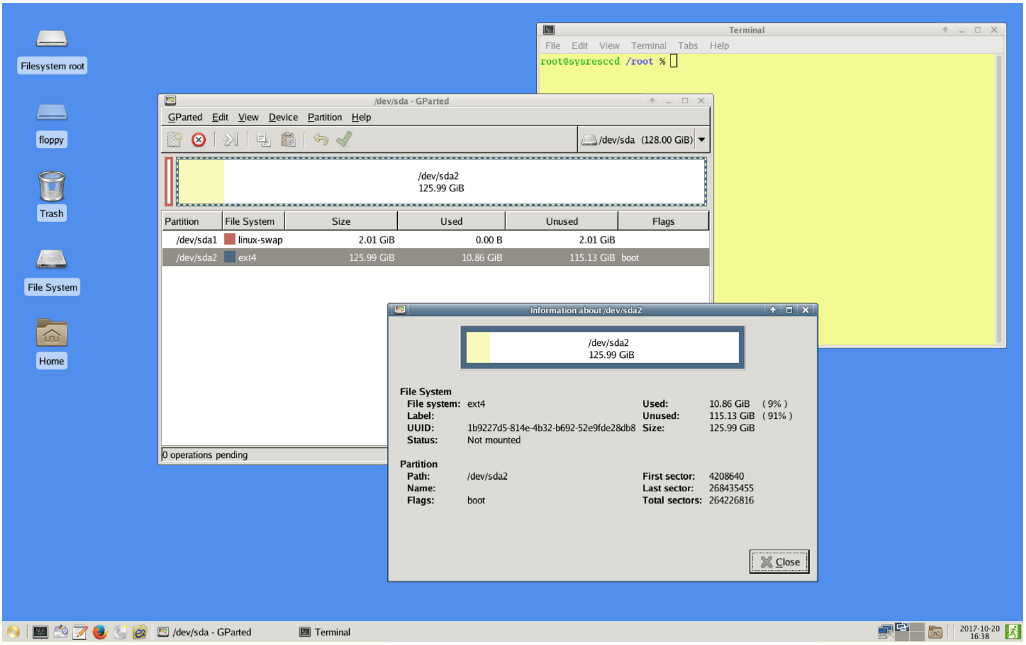
Μετά το φόρτωμα των kernel modules, το SystemRescueCd μάς ζητά να επιλέξουμε keymap για το πληκτρολόγιο. Αν δεν κάνουμε κάτι, είκοσι δευτερόλεπτα αργότερα χρησιμοποιείται το προκαθορισμένο. Είναι το αμερικάνικο και μας κάνει μια χαρά.



O PXE client διαβάζει το αρχείο sysrcd.dat, το οποίο αποτελεί το root filesystem image του SystemRescueCd.



Αφού ολοκληρωθεί η εκκίνηση του SystemRescueCd από πλευράς PXE client, βρισκόμαστε σε μια κονσόλα απλού κειμένου της διανομής. Μπορούμε φυσικά να εργαστούμε από αυτή ή, εναλλακτικά, να πληκτρολογήσουμε startx και να φορτώσουμε το περιβάλλον γραφικών.



Ιδού και το GParted, στο περιβάλλον γραφικών του SystemRescueCd.

Skill: Intermediate
Tags: DNS, proxying, SOCKS, SSH, tunneling

Proxy commands για το SSH και SOCKS tunneling για τον browser

Κάποια εργαλεία είναι τόσο αξιόπιστα και βολικά, ώστε τα χρησιμοποιούμε σχεδόν καθημερινά για να κάνουμε τη δουλειά μας χωρίς καν να το σκεφτόμαστε. Σίγουρα είναι εντελώς αναμενόμενη –κι επιθυμητή– αυτή η στάση, μόνο που μερικές φορές μας εμποδίζει από το να διακρίνουμε εναλλακτικούς και ιδιαίτερα βολικούς τρόπους χρήσης αυτών των ίδιων εργαλείων.

Κατά νου έχουμε τώρα το πρωτόκολλο SSH και συγκεκριμένα το εργαλείο ssh, από το πακέτο OpenSSH (https://www.openssh.com), με το οποίο εγκαθιδρύουμε πανεύκολα ασφαλείς κρυπτογραφημένες συνδέσεις προς απομακρυσμένα μηχανήματα. Θέλουμε πρόσβαση από το σπίτι σ' ένα μηχάνημά μας στη δουλειά, στο πανεπιστήμιο ή οπουδήποτε αλλού; Με δεδομένο ότι το απομακρυσμένο μηχάνημα δεν εμποδίζεται από firewall, ανοίγουμε ένα τερματικό, πληκτρολογούμε κάτι σαν ssh auser@remote.host.tld, και σε ελάχιστα δευτερόλεπτα έχουμε πλήρη πρόσβαση στο λογαριασμό του χρήστη auser, στο remote.host.tld. Ακόμη κι αν το εν λόγω μηχάνημα δεν είναι προσβάσιμο από το Internet, αν παρέχεται κάποια υπηρεσία VPN προς το απομακρυσμένο δίκτυο τότε συνδεόμαστε **πρώτα** σ' αυτή και **μετά** στο μηχάνημα-προορισμό.

Τα πράγματα αρχίζουν να γίνονται ενδιαφέροντα όταν θέλουμε να φτάσουμε σε μηχάνημα **πίσω** από το απομακρυσμένο, εντός τοπικού δικτύου που **δεν** είναι προσβάσιμο από τον έξω κόσμο. Αναλυτικότερα, ο μόνος τρόπος για να βρισκόμαστε σ' αυτό το δίκτυο είναι μέσω αυτού του απομακρυσμένου host, στον οποίο **έχουμε** πρόσβαση.

Πρόσφατα έτυχε να αντιμετωπίσουμε δύο προβλήματα, σχετικά με μια κατάσταση σαν αυτή που μόλις περιγράψαμε. Δώσαμε και στα δύο λύσεις και πιστεύουμε ότι θα υπάρχουν πέντε, έξι ή λίγοι περισσότεροι αναγνώστες που θα θέλουν να μάθουν περισσότερα. Κάπως έτσι προέκυψε το άρθρο που τώρα διαβάζετε.

Απομονωμένο δίκτυο, σχεδόν πάντα βολικό

Φανταστείτε ότι στο Linux workstation στην εργασία σας, έχετε την υπηρεσία του SSH ενεργοποιημένη. Με ή χωρίς σύνδεση στο εταιρικό VPN, από οποιοδήποτε μέρος του πλανήτη —βεβαίως κι από το Διεθνή Διαστημικό Σταθμό— φτάνετε εύκολα στη γραμμή εντολών του εν λόγω workstation. Το hostname του ας είναι isafjordur, το domain στο οποίο ανήκει ας είναι το colder.xyz, ενώ ο χρήστης στο λογαριασμό του οποίου θέλετε να συνδεόσαστε ας έχει username το sub0.

Από κάποιο άλλο μηχάνημά σας με Linux, λοιπόν, σε ένα τερματικό πληκτρολογείτε ssh sub0@isafjordur.colder.xyz και με ασφάλεια συνδεόσαστε στο isafjordur. Αν μάλιστα έχετε μεταφέρει εκεί το δημόσιο κλειδί του τοπικού σας χρήστη, δεν πληκτρολογείτε καν το password του (απομακρυσμένου) sub0 (https://deltahacker.gr/?p=13357). Για τη σύνδεση ίσως πληκτρολογείτε **ακόμη λιγότερα**, π.χ., κάτι σαν ssh isafjordur, επειδή στο αρχείο ~/.ssh/config του τοπικού σας χρήστη έχετε μια ενότητα (SSH alias) που μοιάζει με την ακόλουθη:

```
Host isafjordur
    HostName isafjordur.colder.xyz
    Port 22
    User sub0
```

Όλα καλά, κατανοητά και συνηθισμένα έως εδώ – ή τουλάχιστον έτσι θέλουμε να πιστεύουμε. Τώρα, το isafjordur είναι ένας KVM host, στον οποίο συχνά-πυκνά δημιουργείτε τοπικά εικονικά δίκτυα (libvirt networks, https://deltahacker.gr/?p=17716). Μέσα σ' αυτά εντάσσετε μεμονωμένα QEMU/KVM VMs ή ακόμη κι ολόκληρα clusters, για εκπαιδευτικούς λόγους ή και για (regression) testing **μετά** την εφαρμογή υποψηφίων maintenance updates. Αν το σενάριο που περιγράφουμε άρχισε να μη βγάζει και πολύ νόημα, δεν υπάρχει κανένας λόγος για ανησυχία: το σημαντικό εδώ είναι να έχουμε κατά νου πως στο isafjordur έχετε φτιάξει ένα ή περισσότερα τοπικά δίκτυα, στα VMs των οποίων εσείς και **μόνον** φτάνετε πανεύκολα αρκεί να ξεκινάτε από το isafjordur.

Ας στρέψουμε την προσοχή μας σε ένα libvirt network με όνομα qbxnet, για το τοπικό domain με όνομα qboxes.here. Χάρη στις κατάλληλες προσθήκες στο /etc/resolv.conf του isafjordur, στα μηχανήματα του qbxnet φτάνουμε χωρίς καν να γνωρίζουμε τα IP τους. Παράδειγμα: σε ένα VM του qbxnet με hostname το leap423 (και, παρεμπιπτόντως, με guest OS το openSUSE 42.3), από το isafjordur συνδεόμαστε στο λογαριασμό του χρήστη userson απλά πληκτρολογώντας ssh userson@leap423.

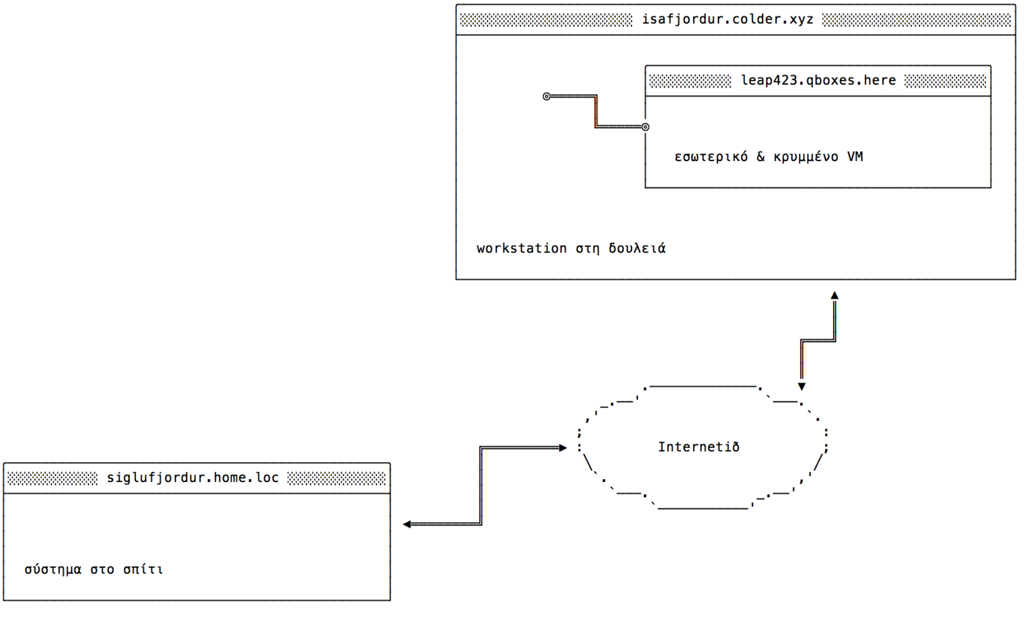
Και κάπου εδώ προκύπτουν μερικά προβλήματα, σαν αυτά που προαναφέραμε. Αναλυτικότερα, αργά ή γρήγορα θα αναρωτηθείτε:

- Πώς από ένα μηχάνημά στο σπίτι συνδεόμαστε **απευθείας** σε μηχάνημα κάποιου απομακρυσμένου εσωτερικού δικτύου, στο οποίο πρόσβαση παρέχεται **μόνον** μέσω κάποιου άλλου –επίσης απομακρυσμένου– host; **Προσοχή**: Η λύση των δύο διαδοχικών συνδέσεων SSH δεν είναι ούτε ιδανική, ούτε επιθυμητή.

- Πώς από ένα μηχάνημα στο σπίτι αποκτάμε πρόσβαση στο περιεχόμενο που προσφέρει web server, ο οποίος τρέχει σε μηχάνημα εντός απομακρυσμένου κι **απομονωμένου** εσωτερικού δικτύου; **Ξανά προσοχή:** Για λόγους απλότητας ή/και ασφαλείας **δεν** επιθυμούμε την εισαγωγή κανόνων port forwarding σε κάποιον άλλον host. Επίσης, για λόγους επιδόσεων **δεν** θέλουμε να τρέξουμε στον απομακρυσμένο host κάποιον web browser, και με X11 forwarding να έχουμε το display στον υπολογιστή στο σπίτι.

Ένα τοπικό δίκτυο σαν το qbxnet είναι σίγουρα βολικό όταν εργαζόμαστε στον physical host που το φιλοξενεί. Σκεφτείτε, για παράδειγμα, ότι μπορούμε να κάνουμε τις δοκιμές μας χωρίς να παράγουμε θόρυβο στο τοπικό δίκτυο που ανήκει το workstation, χωρίς να απασχολούμε DHCP ή/και DNS servers, χωρίς να εισαγάγουμε δυνητικά προβλήματα ασφαλείας κ.ο.κ. Όταν όμως είμαστε σε κάποιο άλλο μηχάνημα, π.χ., στο σπίτι, μήπως το qbxnet είναι κάπως άβολο για να προχωράμε την εργασία μας; Μήπως παραείναι καλά κρυμμένο, για να το πούμε διαφορετικά;

Όχι κατ' ανάγκη. Το τι εννοούμε θα φανεί στη συνέχεια, όπου δίνουμε λύσεις στα δύο προβλήματα που μόλις περιγράψαμε. Ας ξεκινήσουμε πρώτα από το πιο εύκολο.



Από το μηχάνημα siglufjordur, στο σπίτι, θέλουμε να συνδεόμαστε απευθείας σε ένα κρυμμένο VM εντός του απομακρυσμένου host με όνομα isafjordur, στη δουλειά.

SSH proxy command

Συνοψίζουμε το πρόβλημα: Είμαστε στο μηχάνημά μας στο σπίτι, με FQDN το siglufjordur.home.loc, και θέλουμε έναν τρόπο προκειμένου να συνδεόμαστε μέσω SSH **απευθείας** στο λογαριασμό του χρήστη userson, στο μηχάνημα

leap423.qboxes.here. Το leap423 είναι προσβάσιμο **μόνον** από το isafjordur.colder.xyz, στο οποίο **έχουμε** απομακρυσμένη πρόσβαση από το siglufjordur.

Το αρχείο ~/.ssh/config, στον προσωπικό κατάλογο του χρήστη μας στο τοπικό μηχάνημα, φροντίζουμε ώστε να εμπεριέχει τα δύο ακόλουθα SSH aliases:

```
Host isafjordur
   HostName isafjordur.colder.xyz
   Port 22
   User sub0

Host leap423
   ProxyCommand ssh -q isafjordur -W leap423.qboxes.here:22
   User userson
```

Αυτά τα δύο SSH aliases είναι ό,τι χρειάζεται το ssh, ώστε να μας συνδέει στο λογαριασμό του userson στο leap423 **μέσω** του λογαριασμού του sub0 στο isafjordur. Από τη μεριά μας, το μόνο που θα πληκτρολογούμε στον υπολογιστή στο σπίτι είναι ssh leap423. Αν μάλιστα το δημόσιο κλειδί του χρήστη μας είναι στο αρχείο ~/.ssh/authorized_keys του userson, στο leap423, τότε δεν θα χρειάζεται καν να δίνουμε το αντίστοιχο password. Παρατηρήστε εξάλλου το δεύτερο από τα παραπάνω SSH aliases: χάρη στο local DNS resolving που ισχύει στο isafjordur για τα hosts του qboxes.here, απλά καθορίσαμε την ονομαστική διεύθυνση του leap423. Γράψαμε, δηλαδή, το πλήρες FQDN του, που είναι leap423.qboxes.here. Αν από πλευράς isafjordur δεν παρεχόταν DNS resolving για το qboxes.here, τότε αντί για το FQDN θα καθορίζαμε την αριθμητική διεύθυνση IP του leap423 (η οποία στην περίπτωσή μας είναι 10.10.10.230).

Πριν συνεχίσουμε, προσέξτε ξανά τη γραμμή ProxyCommand, στο SSH alias για το leap423. Ίσως το συντακτικό της αρχικά μοιάζει παράξενο, στην πραγματικότητα όμως είναι αρκετά απλό. Στο παράδειγμά μας, σημαίνει: *Συνδέσου μέσω SSH στο TCP port 22 του μηχανήματος leap423.qboxes.here, αφού **πρώτα** πραγματοποιήσεις τη σύνδεση που ορίζεται στο SSH alias ονόματι isafjordur. Α, κι έχε υπόψη ότι στο leap423.qboxes.here θέλουμε σύνδεση στο λογαριασμό του χρήστη με username το userson.*

Συνδυασμός με SOCKS tunneling

Για άλλη μια φορά, συνοψίζουμε το πρόβλημα: Είμαστε στο μηχάνημά μας στο σπίτι, με FQDN το siglufjordur.home.loc, και θέλουμε έναν τρόπο ώστε να βλέπουμε το περιεχόμενο που παρέχει ο web server ενός VM με όνομα s5master.qboxes.here. Το s5master είναι προσβάσιμο **μόνον** από το απομακρυσμένο isafjordur.colder.xyz, στο οποίο **έχουμε** πρόσβαση από το siglufjordur.

Ξεκινάμε από το οικιακό μηχάνημα, το siglufjordur. Στο αρχείο ~/.ssh/config φροντίζουμε ώστε να υπάρχουν οι ακόλουθες δύο ενότητες (SSH aliases):

```
Host isafjordur
HostName isafjordur.colder.xyz
Port 22
User sub0

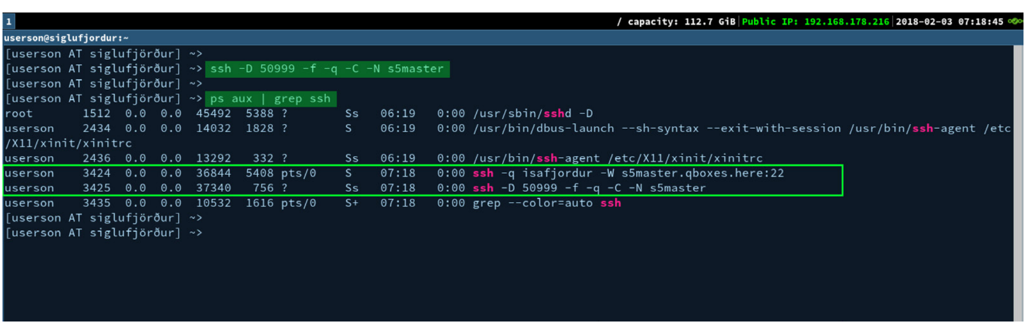
Host s5master
ProxyCommand ssh -q isafjordur -W s5master.qboxes.here:22
User root
```

Πολύ ωραία. Χάρη στα δύο αυτά aliases, από το λογαριασμό μας στο isafjordur συνδεόμαστε στο λογαριασμό του χρήστη root στο s5master, απλά πληκτρολογώντας ssh s5master. Παρεμπιπτόντως, άπαξ και προσθέσουμε το δημόσιο κλειδί του τοπικού μας χρήστη στο αρχείο ~/.ssh/authorized_keys του root στο s5master, τότε κατά την εγκαθίδρυση σύνδεσης δεν θα μας ζητείται καν το password του root.

Έως εδώ έχουμε κάνει **ακριβώς** ό,τι κάναμε για την αντιμετώπιση του προηγούμενου προβλήματος. Το τρέχον πρόβλημα εξακολουθεί να υφίσταται, αλλά μόλις δημιουργήσαμε τις προϋποθέσεις για την επίλυσή του. Πράγματι, σε ένα τερματικό πληκτρολογούμε:

```
ssh -D 50999 -f -q -C -N s5master
```

Αν όλα έγιναν σωστά, τότε στο τερματικό μας δεν βλέπουμε τίποτε απολύτως. Γράφοντας όμως ps aux | grep ssh, το αποτέλεσμα που παίρνουμε μοιάζει με το ακόλουθο και φανερώνει ότι **κάτι** έχει συμβεί.



Εγκαθίδρυση SOCKS tunnel από το τοπικό port 50999/TCP προς τον απομακρυσμένο και απομονωμένο host με όνομα s5master, μέσω του απομακρυσμένου host isafjordur.

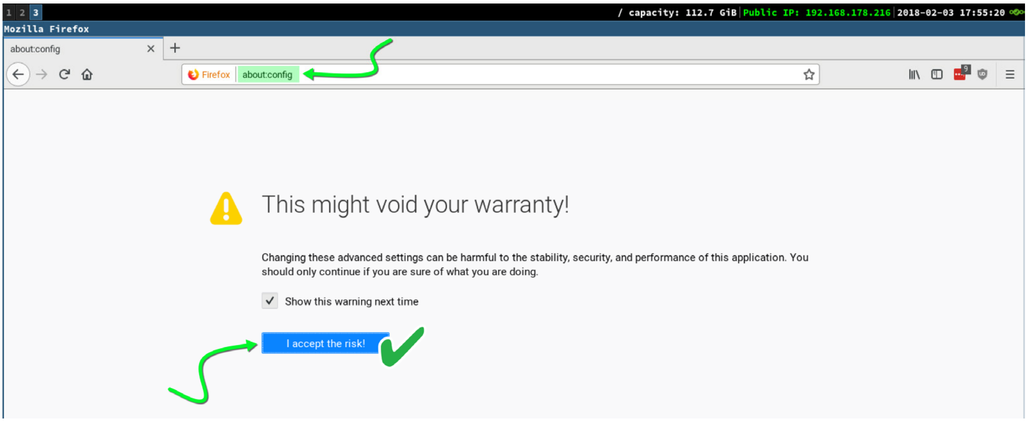
Γίνεται λοιπόν φανερό ότι έχουμε συνδεθεί στο s5master, αλλά ποια είναι η σημασία όλων αυτών των παραμέτρων που δώσαμε στο ssh; Τι πετύχαμε, αν πετύχαμε κάτι; Ας δούμε πρώτα τις παραμέτρους.

- **-D:** Λέμε στο ssh να δημιουργήσει ένα SOCKS tunnel, ξεκινώντας από ένα τοπικό TCP port μεταξύ 1025-65536 (στο παράδειγμά μας καθορίσαμε το port 50999).
- **-f:** Η νέα διεργασία (process) περνά στο υπόβαθρο (background), οπότε μετά την εκτέλεση της εντολής συνεχίζουμε να έχουμε κανονικά πρόσβαση στο τερματικό μας. Ακόμη και να το κλείσουμε, η διεργασία θα συνεχίσει να τρέχει – και συνεπώς το SOCKS tunnel να υφίσταται.
- **-q:** Δεν επιθυμούμε πληροφοριακά μηνύματα στο τερματικό. Τι να τα κάνουμε; (Όχι, σοβαρά τώρα, δεν τα χρειαζόμαστε.)
- **-C:** Ζητάμε τη συμπίεση των δεδομένων **πριν** μπουν στο τούνελ. Κατ' αυτόν τον τρόπο συμβάλλουμε στην καλύτερη αξιοποίηση του bandwidth.
- **-N:** Λέμε στο ssh ότι μετά τη δημιουργία του τούνελ **δεν** θέλουμε να στείλουμε κάποια εντολή στο μηχάνημα στην άλλη άκρη.

Με σχετικά απλό τρόπο μόλις εγκαθιδρύσαμε ένα κρυπτογραφημένο κανάλι επικοινωνίας μεταξύ του τοπικού υπολογιστή κι ενός απομακρυσμένου και κρυμμένου host (s5master), μέσω ενός άλλου απομακρυσμένου host (isafjordur). Αυτή τη στιγμή βέβαια δεν έχουμε πρόσβαση στην κονσόλα του s5master, οπότε ποιο είναι το κέρδος μας; Τέλεια ερώτηση – κι εντελώς αναμενόμενη. Το θέμα εδώ, φίλες και φίλοι, είναι ότι υπάρχουν εφαρμογές που έχουν τη δυνατότητα να χρησιμοποιούν SOCKS tunnels προκειμένου να στέλνουν και να λαμβάνουν δικτυακά πακέτα. Μία απ' αυτές τις εφαρμογές είναι ο Firefox. Αν λοιπόν τον ρυθμίσουμε κατάλληλα, τότε το web browsing από τη συγκεκριμένη εφαρμογή θα γίνεται **μέσω** του s5master. Ο Firefox θα είναι σαν να βγαίνει στο Internet από τον εν λόγω host, άρα θα έχουμε πρόσβαση **και** στο περιεχόμενο που σερβίρει ο web server στο s5master! Χωρίς άλλες καθυστερήσεις, πάμε αμέσως να δούμε πώς ρυθμίζουμε τον Firefox ώστε να χρησιμοποιεί το SOCKS tunnel που πριν λίγο δημιουργήσαμε.

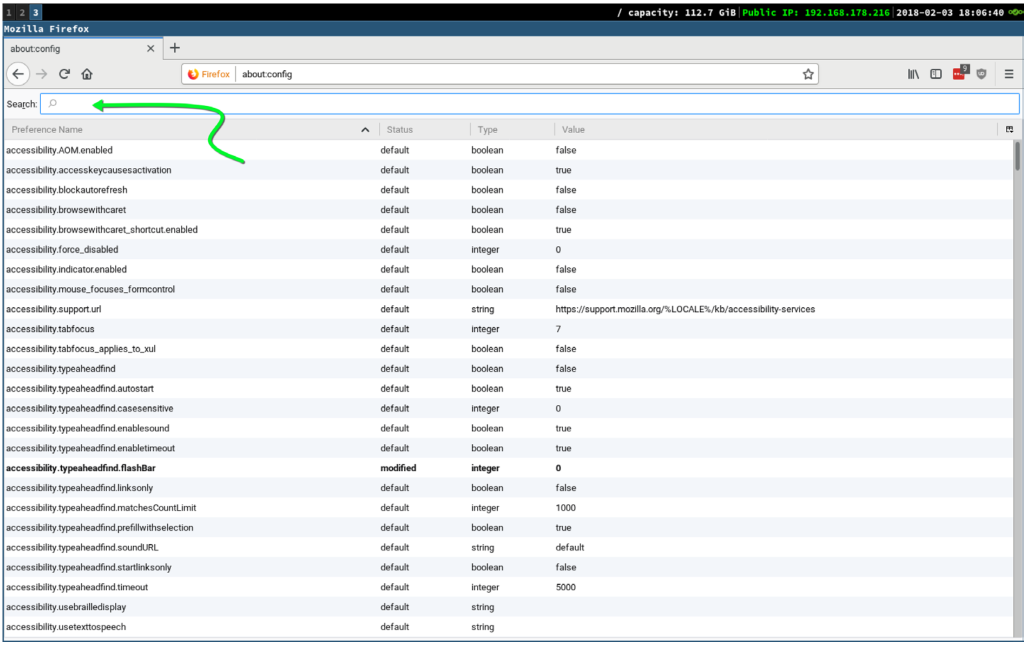
Remote DNS και SOCKS tunneling για τον Firefox

Ξεκινάμε σημειώνοντας ότι, στο πλαίσιο του παρόντος άρθρου, δουλέψαμε με την έκδοση 58.0.1 του Firefox. Αρχικά, στη μπάρα διευθύνσεων του browser πληκτρολογούμε about:config και πατάμε το πλήκτρο [Enter]. Θέλουμε μ' αυτόν τον τρόπο να αποκτήσουμε πρόσβαση στις προχωρημένες (κι εξ ορισμού κρυμμένες) επιλογές της εφαρμογής. Αμέσως θα λάβουμε μια δραματική (και λίγο αστεία) προειδοποίηση, αφού παίζοντας με τις συγκεκριμένες επιλογές είναι πιθανό να κάνουμε περισσότερο κακό παρά καλό. Εμείς βέβαια τώρα ξέρουμε πολύ καλά τι κάνουμε, το ίδιο κι εσείς. Αποφαστιστικά και χωρίς τον παραμικρό δισταγμό, κάνουμε ένα κλικ στο κουμπί I accept the risk.



Παίζοντας με τις προχωρημένες (κι εξ ορισμού κρυμμένες) ρυθμίσεις του Firefox, είναι πιθανό να κάνουμε περισσότερο κακό παρά καλό. Αν πάλι ξέρουμε πολύ καλά τι πάμε να κάνουμε, τότε με αποφασιστικότητα πατάμε σ' αυτό το "I accept the risk".

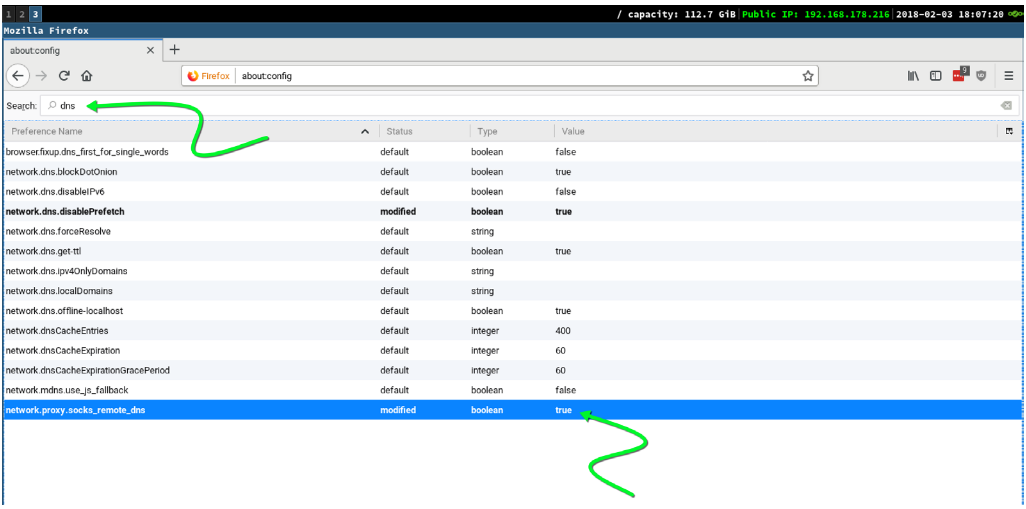
Θα εμφανιστεί μία σελίδα με όλες τις κρυμμένες ρυθμίσεις του Firefox. Επειδή είναι καμιά εβδομηνταριά χιλιάδες (ή λίγο λιγότερες), στη θυρίδα **Search** μπορούμε να πληκτρολογήσουμε μέρος του ονόματος της ρύθμισης κι από κάτω θα εμφανιστούν μόνον οι ρυθμίσεις που στο όνομά τους εμπεριέχουν ό,τι έχουμε γράψει έως αυτή τη στιγμή.



Preference Name	Status	Type	Value
accessibility.AOM.enabled	default	boolean	false
accessibility.accesskeycausesactivation	default	boolean	true
accessibility.blockautorefresh	default	boolean	false
accessibility.browsewithcaret	default	boolean	false
accessibility.browsewithcaret_shortcut.enabled	default	boolean	true
accessibility.force_disabled	default	integer	0
accessibility.indicator.enabled	default	boolean	false
accessibility.mouse_focuses_formcontrol	default	boolean	false
accessibility.support.url	default	string	https://support.mozilla.org/%LOCALE%/kb/accessibility-services
accessibility.tabfocus	default	integer	7
accessibility.tabfocus_applies_to_xul	default	boolean	false
accessibility.typeaheadfind	default	boolean	false
accessibility.typeaheadfind.autostart	default	boolean	true
accessibility.typeaheadfind.casesensitive	default	integer	0
accessibility.typeaheadfind.enablesound	default	boolean	true
accessibility.typeaheadfind.enabletimeout	default	boolean	true
accessibility.typeaheadfind.flashBar	**modified**	**integer**	**0**
accessibility.typeaheadfind.linksonly	default	boolean	false
accessibility.typeaheadfind.matchesCountLimit	default	integer	1000
accessibility.typeaheadfind.prefillwithselection	default	boolean	true
accessibility.typeaheadfind.soundURL	default	string	default
accessibility.typeaheadfind.startlinksonly	default	boolean	false
accessibility.typeaheadfind.timeout	default	integer	5000
accessibility.usebrailledisplay	default	string	
accessibility.usetexttospeech	default	string	

Στη θυρίδα "Search" μπορούμε να πληκτρολογήσουμε μέρος του ονόματος της επιθυμητής ρύθμισης κι από κάτω θα εμφανιστούν μόνον οι ρυθμίσεις που στο όνομά τους εμπεριέχουν ό,τι έχουμε γράψει στη θυρίδα.

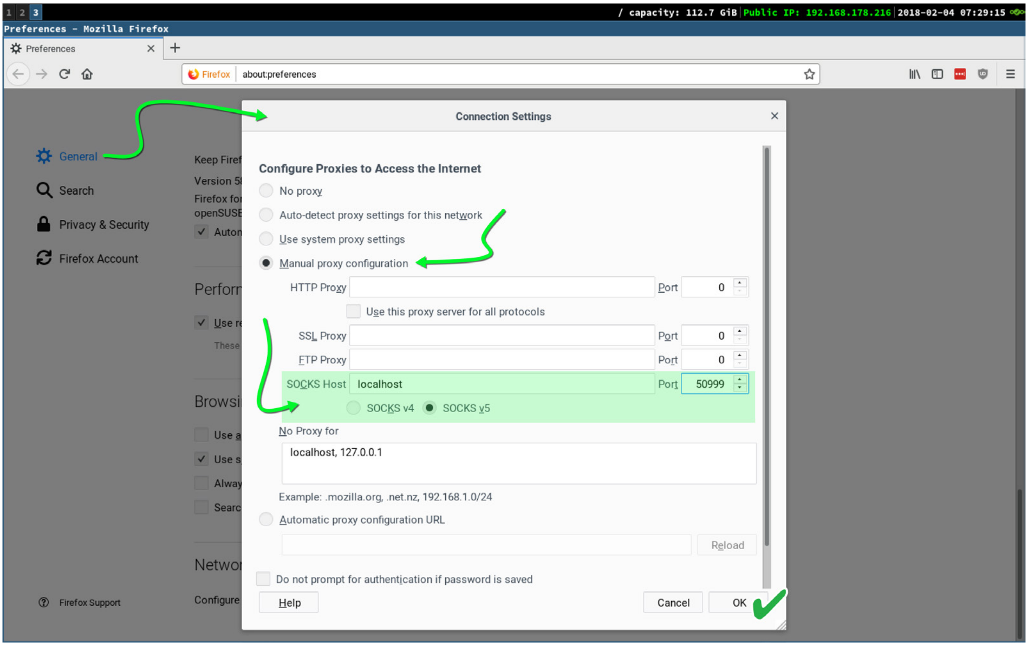
Θέλουμε να ζητήσουμε από τον Firefox να αγνοεί τους τοπικά ορισμένους name servers, όταν χρησιμοποιεί SOCKS proxy, και να προωθεί τα σχετικά ερωτήματα μέσω του τούνελ. Προς τούτο, στη θυρίδα Search γράφουμε dns. Χωρίς να πατήσουμε [Enter], βλέπουμε ότι η λίστα ρυθμίσεων περιορίζεται από μόνη της. Στρέφουμε την προσοχή μας στη ρύθμιση –ή καλύτερα στην παράμετρο– με όνομα network.proxy.socks_remote_dns. Ο τύπος της είναι boolean, που σημαίνει ότι δέχεται μία από δύο τιμές: εν προκειμένω, είτε την true είτε την false. Η τρέχουσα τιμή αλλάζει με κλικ πάνω στη γραμμή της παραμέτρου. Περιττό να σημειώσουμε ότι θέλουμε να ισχύει network.proxy.socks_remote_dns = true.



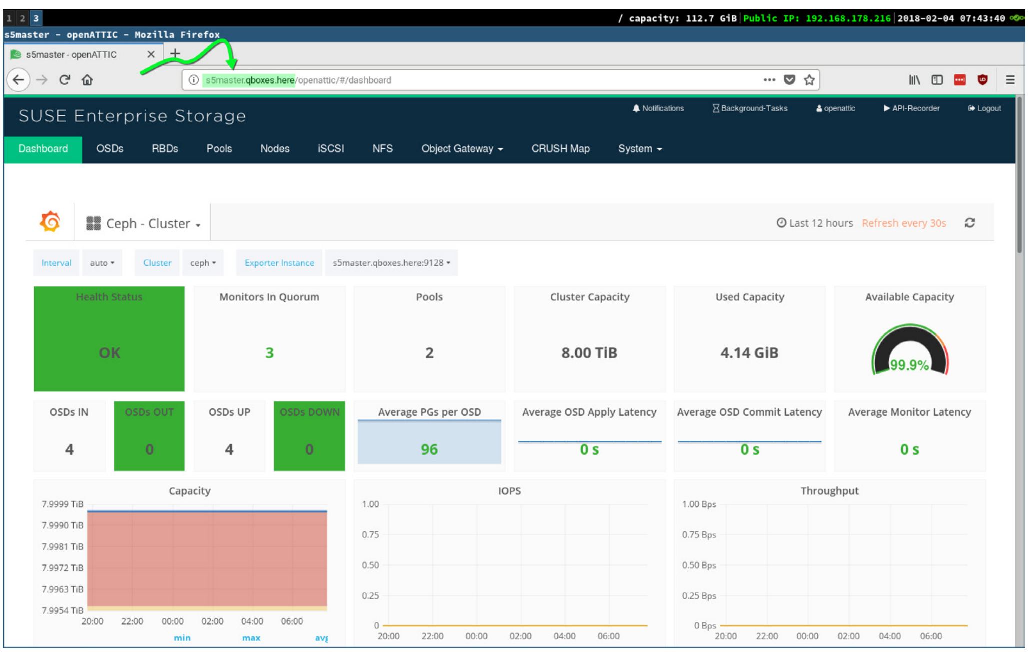
Τα DNS requests του Firefox θέλουμε να αποστέλλονται, μέσω του SOCKS tunnel, στον name server του μηχανήματος στην άλλη άκρη του τούνελ. Γι' αυτό και φροντίζουμε ώστε η επιλεγμένη παράμετρος να είναι "true".

Στη συνέχεια ανοίγουμε την καρτέλα με τις (φανερές) ρυθμίσεις του Firefox. Πάνω κι αριστερά, κάνουμε ένα κλικ στην κατηγορία General. Στο κάτω μέρος της σελίδας, στην ενότητα Network Proxy, πατάμε στο κουμπί Settings. Εμφανίζεται τότε ένα pop-up window, με όνομα Connection Settings. Για το Configure Proxies to Access the Internet επιλέγουμε το Manual proxy configuration. Στη θυρίδα SOCKS Host από κάτω, πληκτρολογούμε localhost. Στη θυρίδα Port, δεξιά, βάζουμε 50999 – ή το port που έχουμε δώσει στην παράμετρο -D του ssh. Για την έκδοση του πρωτοκόλλου SOCKS επιλέγουμε το SOCKS v5. Επικυρώνουμε τις επιλογές μας με ένα κλικ στο κουμπί OK, κάτω δεξιά.

Είμαστε έτοιμοι να δούμε το περιεχόμενο που σερβίρει ο web server στην άλλη άκρη του τούνελ. Για το παράδειγμά μας, στην άλλη άκρη έχουμε το μηχάνημα με FQDN s5master.qboxes.here. Επειδή ρυθμίσαμε τον Firefox ώστε να μη χρησιμοποιεί τους τοπικούς name servers αλλά εκείνους στο απομακρυσμένο άκρο του τούνελ, δεν χρειάζεται να γνωρίζουμε την αριθμητική διεύθυνση του s5master. Στη μπάρα διευθύνσεων του browser, λοιπόν, αρκεί να πληκτρολογήσουμε κάτι σαν http://s5master.qboxes.here (ναι, για το παράδειγμά μας χρησιμοποιούμε το **απλό** HTTP).



Με την προϋπόθεση ότι το SOCKS tunnel προς το απομακρυσμένο host είναι ενεργό, χάρη στις εικονιζόμενες ρυθμίσεις ο Firefox θα χρησιμοποιεί το συγκεκριμένο τούνελ για την επικοινωνία του με τον έξω κόσμο.



Συνδυάζοντας τη δυνατότητα του ssh για Proxy commands και το SOCKS tunneling, από τον υπολογιστή μας στο σπίτι και μέσω τρίτου host αποκτάμε πρόσβαση στο περιεχόμενο που σερβίρει ο web server ενός απομακρυσμένου κι απομονωμένου host.

Στο πλαίσιο της παρουσίασής μας ο web server στο s5master.qboxes.here σερβίρει το web dashboard του openATTIC, για ένα πειραματικό SUSE Enterprise Storage cluster.

SOCKS tunneling χωρίς SSH Proxy Commands

Αρκετές φορές δεν ενδιαφερόμαστε να φτάνουμε σε κάποιο απομακρυσμένο ή/και κρυφό δίκτυο, θέλουμε όμως έναν ασφαλή τρόπο για να βγαίνουμε στο Internet και την ίδια στιγμή να προστατεύσουμε την ιδιωτικότητά μας. Η καλύτερη λύση σε τέτοιες περιπτώσεις ακούει στο όνομα **OpenVPN**. (Διαβάστε, παρεμπιπτόντως, πώς μπορείτε να στήσετε τον δικό σας OpenVPN server σε ένα ταπεινό Raspberry Pi (https://deltahacker.gr/?p=17607). Για να δουλέψει όμως χρειάζεται να έχει προηγηθεί αρκετή δουλειά στη μεριά του OpenVPN server, ενώ πρέπει να έχουμε και το κατάλληλο software σε κάθε client. Αν όμως διαθέτουμε κάποιο μηχάνημα ή VPS με OpenSSH server, το οποίο βεβαίως είναι προσβάσιμο από το Internet, τότε ένα απλό SOCKS tunnel αποτελεί μια έξυπνη και γρήγορη λύση για το πρόβλημά μας. Υπογραμμίζουμε ότι **δεν** θα περνάει όλη η δικτυακή κίνηση του υπολογιστή μέσα από το SOCKS tunnel, αλλά μόνο των εφαρμογών που είναι ικανές να χρησιμοποιούν το τούνελ και φυσικά τις έχουμε ρυθμίσει ώστε να το κάνουν. Στις περισσότερες των περιπτώσεων, η μόνη εφαρμογή που θα θέλουμε να ρυθμίζουμε είναι ο web browser. Επίσης, για ένα τέτοιο σενάριο είναι μάλλον απίθανο να συνδεόμαστε στο απομακρυσμένο μηχάνημα μέσω τρίτου μηχανήματος. Μ' άλλα λόγια, δεν θα υπάρχει η ανάγκη για τα proxy commands του SSH: απλά θα γράφουμε κάτι σαν ssh -D 50999 -f -q -C -N userson@isafjordur.colder.xyz κι αφού το τούνελ εγκαθιδρύεται θα ρυθμίζουμε τον Firefox ώστε να χρησιμοποιεί το localhost:50999 ως SOCKS5 proxy.

Καταστροφή υφιστάμενου SOCKS tunnel

Όταν δεν χρειαζόμαστε άλλο ένα SOCKS tunnel, καλό είναι να το καταστρέφουμε αμέσως. Πληκτρολογώντας στο τερματικό μας ps aux | grep ssh παίρνουμε μια λίστα με τις διεργασίες που έχουν σχέση με το ssh. Αυτή που μας ενδιαφέρει, στο δεξιό της μέρος θα έχει μια εντολή που ξεκινά κάπως έτσι: ssh -D 50999. Το πρώτο πεδίο από αριστερά είναι το όνομα του χρήστη ο οποίος ξεκίνησε τη διεργασία. Ακριβώς στα δεξιά του είναι το πεδίο με το PID της διεργασίας. Αν, π.χ., το PID είναι 7914, τότε για την καταστροφή του τούνελ αρκεί να πληκτρολογήσουμε kill 7914. Επιβεβαιώνουμε πως όλα πήγαν κατ' ευχήν δίνοντας ξανά ps aux | grep ssh. Τέλος, δεν πρέπει να ξεχνάμε και τον Firefox. Απλά πηγαίνουμε στις ρυθμίσεις του και φροντίζουμε ώστε να μη χρησιμοποιείται το SOCKS tunnel ως proxy.

deltahacker.gr